

Metric tree-like structures in real-life networks: an empirical study

Muad Abu-Ata and Feodor F. Dragan

Algorithmic Research Laboratory, Department of Computer Science
Kent State University, Kent, OH 44242, USA
{*mabuata, dragan*}@cs.kent.edu

Abstract. Based on solid theoretical foundations, we present strong evidences that a number of real-life networks, taken from different domains like Internet measurements, biological data, web graphs, social and collaboration networks, exhibit tree-like structures from a metric point of view. We investigate few graph parameters, namely, the tree-distortion and the tree-stretch, the tree-length and the tree-breadth, the Gromov’s hyperbolicity, the cluster-diameter and the cluster-radius in a layering partition of a graph, which capture and quantify this phenomenon of being metrically close to a tree. By bringing all those parameters together, we not only provide efficient means for detecting such metric tree-like structures in large-scale networks but also show how such structures can be used, for example, to efficiently and compactly encode approximate distance and almost shortest path information and to fast and accurately estimate diameters and radii of those networks. Estimating the diameter and the radius of a graph or distances between its arbitrary vertices are fundamental primitives in many data and graph mining algorithms.

1 Introduction

Large networks are everywhere. Can we understand their structure and exploit it? For example, understanding key structural properties of large-scale data networks is crucial for analyzing and optimizing their performance, as well as improving their reliability and security [56]. In prior empirical and theoretical studies researchers have mainly focused on features like small world phenomenon, power law degree distribution, navigability, high clustering coefficients, etc. (see [8,9,11,26,39,?,51,54,65]). Those nice features were observed in many real-life complex networks and graphs arising in Internet applications, in biological and social sciences, in chemistry and physics. Although those features are interesting and important, as it is noted in [56], the impact of intrinsic geometrical and topological features of large-scale data networks on performance, reliability and security is of much greater importance.

Recently, a few papers explored a little-studied before geometric characteristic of real-life networks, namely the *hyperbolicity* (sometimes called also the *global curvature*) of the network (see, e.g., [4,20,28,48,56,62]). It was shown that a number of data networks, including Internet application networks, web networks, collaboration networks, social networks, and others, have small hyperbolicity. It was suggested in [56] that property, observed in real-life networks, that traffic between nodes tends to go through a relatively small core of the network, as if the shortest path between them is curved inwards, may be due to global curvature of the network. Furthermore, paper [48] proposes that “hyperbolicity in conjunction with other local characteristics of networks, such as the degree distribution and clustering coefficients, provide a more complete unifying picture of networks, and helps classify in a parsimonious way what is otherwise a bewildering and complex array of features and characteristics specific to each natural and man-made network”.

The hyperbolicity of a graph/network can be viewed as a measure of how close a graph is to a tree metrically; the smaller the hyperbolicity of a graph is the closer it is metrically to a tree. Recent empirical results of [4,20,28,48,56,62] on hyperbolicity suggest that many real-life complex networks and graphs may possess tree-like structures from a metric point of view.

In this paper, we substantiate this claim through analysis of a collection of real data networks. We investigate few more, recently introduced graph parameters, namely, the *tree-distortion* and the *tree-stretch* of a graph, the *tree-length* and the *tree-breadth* of a graph, the Gromov’s *hyperbolicity* of a graph, the *cluster-diameter* and the *cluster-radius* in a *layering partition* of a graph. All these parameters are trying to capture

and quantify this phenomenon of being metrically close to a tree and can be used to measure metric tree-likeness of a real-life network. Recent advances in theory (see appropriate sections for details) allow us to calculate or accurately estimate those parameters for sufficiently large networks. By examining topologies of numerous publicly available networks, we demonstrate existence of metric tree-like structures in wide range of large-scale networks, from communication networks to various forms of social and biological networks.

Throughout this paper we discuss these parameters and recently established relationships between them for unweighted and undirected graphs. It turns out that all these parameters are at most constant or logarithmic factors apart from each other. Hence, a constant bound on one of them translates in a constant or almost constant bound on another. We say that a graph *has a tree-like structure from a metric point of view* (equivalently, *is metrically tree-like*) if anyone of those parameters is a small constant.

Recently, paper [4] pointed out that "although large informatics graphs such as social and information networks are often thought of as having hierarchical or tree-like structure, this assumption is rarely tested, and it has proven difficult to exploit this idea in practice; ... it is not clear whether such structure can be exploited for improved graph mining and machine learning ...".

In this paper, by bringing all those parameters together, we not only provide efficient means for detecting such metric tree-like structures in large-scale networks but also show how such structures can be used, for example, to efficiently and compactly encode approximate distance and almost shortest path information and to fast and accurately estimate diameters and radii of those networks. Estimating accurately and quickly distances between arbitrary vertices of a graph is a fundamental primitive in many data and graph mining algorithms.

Graphs that are metrically tree-like have many algorithmic advantages. They allow efficient approximate solutions for a number of optimization problems. For example, they admit a PTAS for the Traveling Salesman Problem [53], have an efficient approximate solution for the problem of covering and packing by balls [25], admit additive sparse spanners [23,32] and collective additive tree-spanners [35], enjoy efficient and compact approximate distance [23,41] and routing [23,31] labeling schemes, have efficient algorithms for fast and accurate estimations of diameters and radii [22], etc.. We elaborate more on these results in appropriate sections.

For the first time such metric parameters, as tree-length and tree-breadth, tree-distortion and tree-stretch, cluster-diameter and cluster-radius, were examined, and algorithmic advantages of having those parameters bounded by small constants were discussed for such a wide range of large-scale networks.

This paper is structured as follows. In Section 2, we give notations and basic notions used in the paper. In Section 3, we describe our graph datasets. The next four sections are devoted to analysis of corresponding parameters measuring metric tree-likeness of our graph datasets: layering partition and its cluster-diameter and cluster-radius in Section 4; hyperbolicity in Section 5; tree-distortion in Section 6; tree-breadth, tree-length and tree-stretch in Section 7. In each section we first give theoretical background on the parameter(s) and then present our experimental results. Additionally, an overview of implications of those results is provided. In Section 8, we further discuss algorithmic advantages for a graph to be metrically tree-like. Finally, in Section 9, we give some concluding remarks.

2 Notations and Basic Notions

All graphs in this paper are connected, finite, unweighted, undirected, loopless and without multiple edges. For a graph $G = (V, E)$, we use n and $|V|$ interchangeably to denote the number of vertices in G . Also, we use m and $|E|$ to denote the number of edges. The *length of a path* from a vertex v to a vertex u is the number of edges in the path. The *distance* $d_G(u, v)$ between vertices u and v is the length of the shortest path connecting u and v in G . The *ball* $B_r(s, G)$ of a graph G centered at vertex $s \in V$ and with radius r is the set of all vertices with distance no more than r from s (i.e., $B_r(s, G) = \{v \in V : d_G(v, s) \leq r\}$). We omit the graph name G as in $B_r(s)$ if the context is about only one graph.

The *diameter* $diam(G)$ of a graph $G = (V, E)$ is the largest distance between a pair of vertices in G , i.e., $diam(G) = \max_{u, v \in V} d_G(u, v)$. The *eccentricity* of a vertex v , denoted by $ecc(v)$, is the largest distance from that vertex v to any other vertex, i.e., $ecc(v) = \max_{u \in V} d_G(v, u)$. The *radius* $rad(G)$ of a graph

$G = (V, E)$ is the minimum eccentricity of a vertex in G , i.e., $rad(G) = \min_{v \in V} \max_{u \in V} d_G(v, u)$. The *center* $C(G) = \{c \in V : ecc(c) = rad(G)\}$ of a graph $G = (V, E)$ is the set of vertices with minimum eccentricity.

Definitions of graph parameters measuring metric tree-likeness of a graph, as well as notions and notations local to a section, are given in appropriate sections.

3 Datasets

Our datasets come from different domains like Internet measurements, biological datasets, web graphs, social and collaboration networks. Table 1 shows basic statistics of our graph datasets. Each graph represents the largest connected component of the original graph as some datasets consist of one large connected component and many very small ones.

Graph $G = (V, E)$	n= $ V $	m= $ E $	diameter $diam(G)$	radius $rad(G)$
PPI [46]	1458	1948	19	11
Yeast [14]	2224	6609	11	6
DutchElite [29]	3621	4311	22	12
EPA [1]	4253	8953	10	6
EVA [57]	4475	4664	18	10
California [49]	5925	15770	13	7
Erdős [10]	6927	11850	4	2
Routeview [2]	10515	21455	10	5
Homo release 3.2.99 [63]	16711	115406	10	5
AS_Caida_20071105 [18]	26475	53381	17	9
Dimes 3/2010 [61]	26424	90267	8	4
Aqualab 12/2007- 09/2008 [19]	31845	143383	9	5
AS_Caida_20120601 [16]	41203	121309	10	5
itdk0304 [17]	190914	607610	26	14
DBLB-coauth [67]	317080	1049866	23	12
Amazon [67]	334863	925872	47	24

Table 1: Graph datasets and their parameters: number of vertices, number of edges, diameter, radius.

Biological Networks

PPI [46]: It is a protein-protein interaction network in the yeast *Saccharomyces Cerevisiae*. Each node represents a protein with an edge representing an interaction between two proteins. Self loops have been removed from the original dataset. The dataset has been analyzed and described in [46].

Yeast [14]: It is a protein-protein interaction network in budding yeast. Each node represents a protein with an edge representing an interaction between two proteins. Self loops have been removed from the original dataset. The dataset has been analyzed and described in [14].

Homo [63]: It is a dataset of protein and genetic interactions in *Homo Sapiens* (Human). Each node represents a protein or a gene. An edge represents an interaction between two proteins/genes. Parallel edges, representing different resources for an interaction, have been removed. The dataset is obtained from BioGRID, a freely accessible database/repository of physical and genetic interactions available at <http://www.thebiogrid.org>. The dataset has been analyzed and described in [63].

Social and Collaboration Networks

DutchElite [29]: This is data on the administrative elite in Netherland, April 2006. Data collected and analyzed by De Volkskrant and Wouter de Nooy. A 2-mode network data representing person's membership in the administrative and organization bodies in Netherland in 2006. A node represents either a person or an organization body. An edge exists between two nodes if the person node belongs to the organization node.

EVA [57]: It is a network of interconnection between corporations where an edge exists between two companies (vertices) if one of them is the owner of the other company.

Erdős [10]: It is a collaboration network with mathematician Paul Erdős. Each vertex represents an author with an edge representing a paper co-authorship between two authors.

DBLP-coauth [67]: It is a co-authorship network of the DBLP computer science bibliography. Vertices of the network represent authors with edges connecting two authors if they published at least one paper together.

Web Graphs

EPA [1]: It is a dataset representing pages linking to www.epa.gov obtained from Jon Kleinberg’s web page, <http://www.cs.cornell.edu/courses/cs685/2002fa/>. The pages were constructed by expanding a 200-page response set to a search engine query, as in the hub/authority algorithm. This data was collected some time back, so a number of the links may not exist anymore. The vertices of this graph dataset represent web pages with edges representing links. The graph was originally directed. We ignored direction of edges to get undirected graph version of the dataset.

California [49]: This graph dataset was also constructed by expanding a 200-page response set to a search engine query ‘California’, as in the hub/authority algorithm. The dataset was obtained from Jon Kleinberg’s page, <http://www.cs.cornell.edu/courses/cs685/2002fa/>. The vertices of this graph dataset represent web pages with edges representing links between them. The graph was originally directed. We ignored direction of edges to obtain undirected graph version of the dataset.

Internet Measurements Networks

Routeview [2]: It is an Autonomous System (AS) graph obtained by University of Oregon Route-views project using looking glass data and routing registry. A vertex in the dataset represents an AS with an edge linking two vertices if there is at least one physical link between them.

AS_Caida [18,16]: These are datasets of the Internet Autonomous Systems (AS) relationships derived from BGP table snapshots taken at 24-hour intervals over a 5-day period by CAIDA. The AS relationships available are customer-provider (and provider-customer, in the opposite direction), peer-to-peer, and sibling-to-sibling.

Dimes 3/2010 [61]: It is an AS relationship graph of the Internet obtained from Dimes. The Dimes project performs traceroutes and pings from volunteer agents (of about 1000 agent computers) to infer AS relationships. A weekly AS snapshot is available. The dataset *Dimes 3/2010* represents a snapshot aggregated over the month of March, 2010. It provides the set of AS level nodes and edges that were found in that month and were seen at least twice.

Aqualab [19]: Peer-to-peer clients are used to collect traceroute paths which are used to infer AS interconnections. Probes were made between December 2007 and September 2008 from approximately 992,000 P2P users in 3,700 ASes.

ItDK [17]: This is a dataset of Internet router-level graph where each vertex represents a router with an edge between two vertices if there is a link between the corresponding routers. The dataset snapshot is computed from ITDK0304 skitter and iffinder measurements. The dataset is provided by CAIDA for April 2003 (see <http://www.caida.org/data/active/internet-topology-data-kit>).

Information network

Amazon [67]: It is an Amazon product co-purchasing network. The vertices of the network represent products purchased from the Amazon website and the edges link “commonly/frequently” co-purchased products.

4 Layering Partition, its Cluster-Diameter and Cluster-Radius

Layering partition is a graph decomposition procedure that has been introduced in [12,21] and has been used in [12,21,24] and [7] for embedding graph metrics into trees. It provides a central tool in our investigation.

A *layering* of a graph $G = (V, E)$ with respect to a start vertex s is the decomposition of V into the layers (spheres) $L^i = \{u \in V : d_G(s, u) = i\}$, $i = 0, 1, \dots, r$. A *layering partition* $\mathcal{LP}(G, s) = \{L_1^i, \dots, L_{p_i}^i : i = 0, 1, \dots, r\}$ of G is a partition of each layer L^i into clusters $L_1^i, \dots, L_{p_i}^i$ such that two vertices $u, v \in L^i$ belong to the same cluster L_j^i if and only if they can be connected by a path outside the ball $B_{i-1}(s)$ of radius $i - 1$ centered at s . See Fig. 1 for an illustration. A layering partition of a graph can be constructed in $O(n + m)$ time (see [21]).

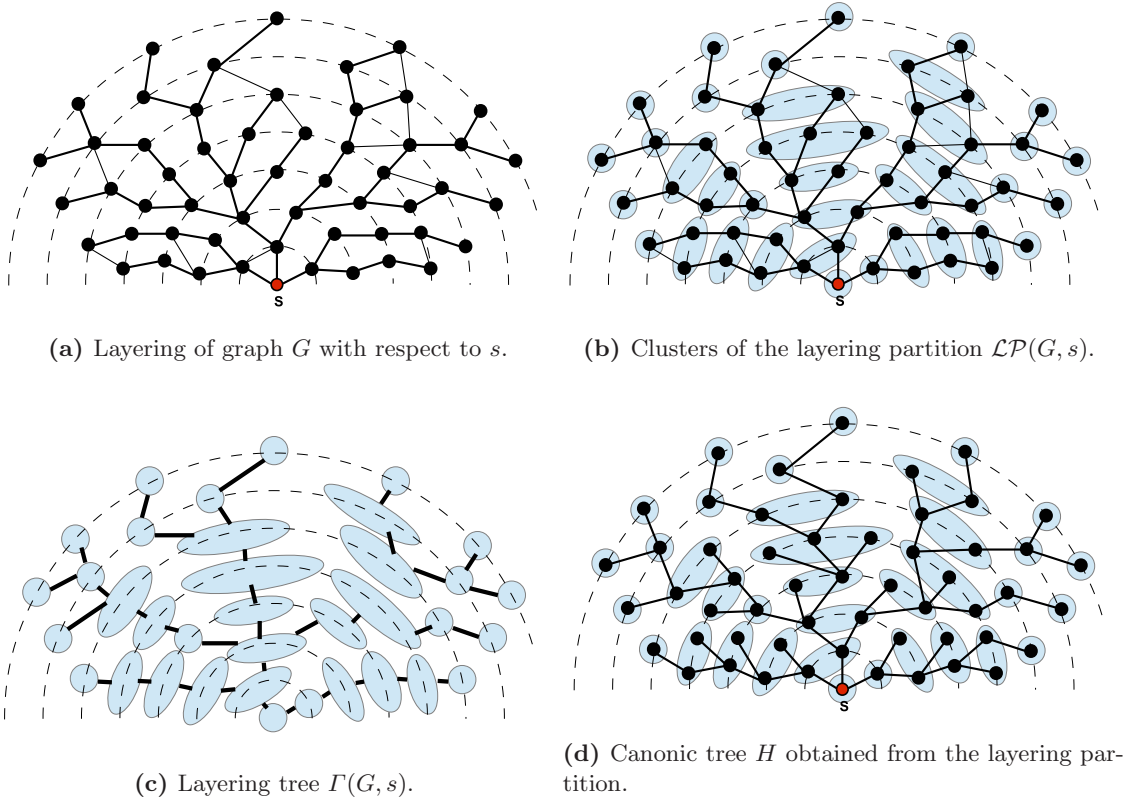


Fig. 1: Layering partition and associated constructs.

A *layering tree* $\Gamma(G, s)$ of a graph G with respect to a layering partition $\mathcal{LP}(G, s)$ is the graph whose nodes are the clusters of $\mathcal{LP}(G, s)$ and two nodes $C = L_j^{i_j}$ and $C' = L_{j'}^{i_{j'}}$ are adjacent in $\Gamma(G, s)$ if and only if there exist a vertex $u \in C$ and a vertex $v \in C'$ such that $uv \in E$. It was shown in [12] that the graph $\Gamma(G, s)$ is always a tree and, given a start vertex s , can be constructed in $O(n + m)$ time [21]. Note that, for a fixed start vertex $s \in V$, the layering partition $\mathcal{LP}(G, s)$ of G and its tree $\Gamma(G, s)$ are unique.

The *cluster-diameter* $\Delta_s(G)$ of layering partition $\mathcal{LP}(G, s)$ with respect to vertex s is the largest diameter of a cluster in $\mathcal{LP}(G, s)$, i.e., $\Delta_s(G) = \max_{C \in \mathcal{LP}(G, s)} \max_{u, v \in C} d_G(u, v)$. The *cluster-diameter* $\Delta(G)$ of a graph G is the minimum cluster-diameter over all layering partitions of G , i.e., $\Delta(G) = \min_{s \in V} \Delta_s(G)$.

The *cluster-radius* $R_s(G)$ of layering partition $\mathcal{LP}(G, s)$ with respect to a vertex s is the smallest number r such that for any cluster $C \in \mathcal{LP}(G, s)$ there is a vertex $v \in V$ with $C \subseteq B_r(v)$. The *cluster-radius* $R(G)$ of a graph G is the minimum cluster-radius over all layering partitions of G , i.e., $R(G) = \min_{s \in V} R_s(G)$.

Clearly, in view of tree $\Gamma(G, s)$ of G , the smaller parameters $\Delta_s(G)$ and $R_s(G)$ of G are, the closer graph G is to a tree metrically.

Finding cluster-diameter $\Delta_s(G)$ and cluster-radius $R_s(G)$ for a given layering partition $\mathcal{LP}(G, s)$ of a graph G requires $O(nm)$ time¹, although the construction of layering partition $\mathcal{LP}(G, s)$ itself, for a given vertex s , takes only $O(n + m)$ time. Since the diameter of any set is at least its radius and at most twice its radius, we have the following inequality:

$$R_s(G) \leq \Delta_s(G) \leq 2R_s(G).$$

In Table 2, we show empirical results on layering partitions obtained for datasets described in Section 3. For each graph dataset $G = (V, E)$, we randomly selected a start vertex s and built layering partition $\mathcal{LP}(G, s)$ of G with respect to s . For each dataset, Table 2 shows the cluster-diameter $\Delta_s(G)$, the number of clusters in layering partition $\mathcal{LP}(G, s)$ and the average diameter of clusters in $\mathcal{LP}(G, s)$. It turns out that

¹ The parameters $\Delta(G)$ and $R(G)$ can also be computed in total $O(nm)$ time for any graph G .

Graph $G = (V, E)$	$n = V $	diameter $diam(G)$	# of clusters in $\mathcal{LP}(G, s)$	cluster- diameter $\Delta_s(G)$	average diameter of clusters in $\mathcal{LP}(G, s)$	% of clusters having diameter 0 or 1 (i.e., cliques)
PPI	1458	19	1017	8	0.118977384	97.05014749%
Yeast	2224	11	1838	6	0.119575699	96.33558341%
DutchElite	3621	22	2934	10	0.070211316	98.02317655%
EPA	4253	10	2523	6	0.06698375	98.5731272%
EVA	4475	18	4266	9	0.031879981	99.2030005%
California	5925	13	2939	8	0.092208234	97.141885%
Erdős	6927	4	6288	4	0.001113232	99.9681934%
Routeview	10515	10	6702	6	0.063264697	98.4482244%
Homo release 3.2.99	16711	10	6817	5	0.03432595	99.2518703%
AS_Caida_20071105	26475	17	17067	6	0.056424679	98.5527626%
Dimes 3/2010	26424	8	16065	4	0.056582633	98.5434174%
Aqualab 12/2007- 09/2008	31845	9	16287	6	0.05826733	98.5816909%
AS_Caida_20120601	41203	10	26562	6	0.055568105	98.5731496%
itdk0304	190914	26	89856	11	0.270377048	91.3851051%
DBLB-coauth	317080	23	99828	11	0.45350002	92.97091%
Amazon	334863	47	72278	21	0.489056144	86.049697%

Table 2: Layering partitions of the datasets and their parameters. $\Delta_s(G)$ is the largest diameter of a cluster in $\mathcal{LP}(G, s)$, where s is a randomly selected start vertex. For all datasets, the average diameter of a cluster is between 0 and 1. For most datasets, more than 95% of clusters are cliques.

all graph datasets have small average diameter of clusters. Most clusters have diameter 0 or 1, i.e., they are essentially cliques (=complete subgraphs) of G . For most datasets, more than 95% of clusters are cliques.

To have a better picture on the overall distribution of diameters of clusters, in Table 3, we show the frequencies of diameters of clusters for three sample datasets: PPI, Yeast, and AS_Caida_20071105. It is interesting to note that, in all datasets, the clusters with large diameters induce a connected subtree in the tree $\Gamma(G, s)$. For example, in PPI, the cluster with diameter 8 is adjacent in $\Gamma(G, s)$ to all clusters with diameters 6 and 5. This may indicate that all those clusters are part of the well connected network core.

diameter of a cluster	frequency	relative frequency	diameter of a cluster	frequency	relative frequency	diameter of a cluster	frequency	relative frequency
0	966	0.9499	0	981	0.946	0	16459	0.9644
1	21	0.0206	1	18	0.0174	1	361	0.0216
2	14	0.0138	2	23	0.0223	2	174	0.0102
3	5	0.0049	3	6	0.0058	3	46	0.0027
4	5	0.0049	4	5	0.0048	4	21	0.0012
5	1	0.0001	5	2	0.0019	5	4	0.0002
6	4	0.0039	6	2	0.0019	6	2	0.0001
7	0	0						
8	1	0.0001						

(a) PPI
(b) Yeast
(c) AS_Caida_20071105

Table 3: Frequency of diameters of clusters in layering partition $\mathcal{LP}(G, s)$ (three datasets).

Most of the graph parameters discussed in this paper could be related to a special tree H introduced in [24] and produced from a layering partition of a graph G .

Canonic tree H: A tree $H = (V, F)$ of a graph $G = (V, E)$, called a *canonic tree of G* , is constructed from a layering partition $\mathcal{LP}(G, s)$ of G by identifying for each cluster $C = L_j^i \in \mathcal{LP}(G, s)$ an arbitrary vertex $x_C \in L_{i-1}$ which has a neighbor in $C = L_j^i$ and by making x_C adjacent in H with all vertices $v \in C$

(see Fig. 1d for an illustration). Vertex x_C is called the support vertex for cluster $C = L_j^i$. It was shown in [24] that tree H for a graph G can be constructed in $O(n + m)$ total time.

The following statement from [24] relates the cluster-diameter of a layering partition of G with embedability of graph G into the tree H .

Proposition 1 ([24]). *For every graph $G = (V, E)$ and any vertex s of G ,*

$$\forall x, y \in V, \quad d_H(x, y) - 2 \leq d_G(x, y) \leq d_H(x, y) + \Delta_s(G).$$

The above proposition shows that the distortion of embedding of a graph G into tree H is additively bounded by $\Delta_s(G)$, the largest diameter of a cluster in a layering partition of G . This result confirms that the smaller cluster-diameter $\Delta_s(G)$ (cluster-radius $R_s(G)$) of G is, the closer graph G is to a tree metric. Note that trees have cluster-diameter and cluster-radius equal to 0. Results similar to Proposition 1 were used in [12] to embed a chordal graph to a tree with an additive distortion at most 2, in [21] to embed a k -chordal graph to a tree with an additive distortion at most $k/2 + 2$, and in [24] to obtain a 6-approximation algorithm for the problem of optimal non-contractive embedding of an unweighted graph metric into a weighted tree metric. For every *chordal graph* G (a graph whose largest induced cycles have length 3), $\Delta_s(G) \leq 3$ and $R_s(G) \leq 2$ hold [12]. For every *k -chordal graph* G (a graph whose largest induced cycles have length k), $\Delta_s(G) \leq k/2 + 2$ holds [21]. For every graph G embeddable non-contractively into a (weighted) tree with multiplication distortion α , $\Delta_s(G) \leq 3\alpha$ holds [24]. See Section 6 for more on this topic.

Graph $G = (V, E)$	n= $ V $	m= $ E $	$\delta(G)$
PPI	1458	1948	3.5
Yeast	2224	6609	2.5
DutchElite	3621	4311	4
EPA	4253	8953	2.5
EVA	4475	4664	1
California	5925	15770	3
Erdős	6927	11850	2
Routeview	10515	21455	2.5
Homo release 3.2.99	16711	115406	2
AS_Caida_20071105	26475	53381	2.5
Dimes 3/2010	26424	90267	2
Aqualab 12/2007- 09/2008	31845	143383	2
AS_Caida_20120601	41203	121309	2

Table 4: δ -hyperbolicity of the graph datasets.

5 Hyperbolicity

δ -Hyperbolic metric spaces have been defined by M. Gromov [44] in 1987 via a simple 4-point condition: for any four points u, v, w, x , the two larger of the distance sums $d(u, v) + d(w, x)$, $d(u, w) + d(v, x)$, $d(u, x) + d(v, w)$ differ by at most 2δ . They play an important role in geometric group theory, geometry of negatively curved spaces, and have recently become of interest in several domains of computer science, including algorithms and networking. For example, (a) it has been shown empirically in [62] (see also [3]) that the Internet topology embeds with better accuracy into a hyperbolic space than into an Euclidean space of comparable dimension, (b) every connected finite graph has an embedding in the hyperbolic plane so that the greedy routing based on the virtual coordinates obtained from this embedding is guaranteed to work (see [52]). A connected graph $G = (V, E)$ equipped with standard graph metric d_G is δ -hyperbolic if the metric space (V, d_G) is δ -hyperbolic.

More formally, let G be a graph and u, v, w and x be its four vertices. Denote by S_1, S_2, S_3 the three distance sums, $d_G(u, v) + d_G(w, x)$, $d_G(u, w) + d_G(v, x)$ and $d_G(u, x) + d_G(v, w)$ sorted in non-decreasing

order $S_1 \leq S_2 \leq S_3$. Define the *hyperbolicity of a quadruplet* u, v, w, x as $\delta(u, v, w, x) = \frac{S_3 - S_2}{2}$. Then the *hyperbolicity* $\delta(G)$ of a graph G is the maximum hyperbolicity over all possible quadruplets of G , i.e.,

$$\delta(G) = \max_{u, v, w, x \in V} \delta(u, v, w, x).$$

δ -Hyperbolicity measures the local deviation of a metric from a tree metric; a metric is a tree metric if and only if it has hyperbolicity 0. Note that chordal graphs, mentioned in Section 4, have hyperbolicity at most 1 [13], while k -chordal graphs have hyperbolicity at most $k/4$ [66].

In Table 4, we show the hyperbolicities of most of our graph datasets. The computation of hyperbolicities is a costly operation. We did not compute it for only three very large graph datasets since it would take very long time to calculate. The best known algorithm to calculate hyperbolicity has time complexity of $O(n^{3.69})$, where n is the number of vertices in the graph; it was proposed in [40] and involves matrix multiplications. This algorithm still takes long running time for large graphs and is hard to implement. Authors of [40] also propose a 2-approximation algorithm for calculating hyperbolicity that runs in $O(n^{2.69})$ time and a $2 \log_2 n$ -approximation algorithm that runs in $O(n^2)$ time. In our computations, we used the naive algorithm which calculates the exact hyperbolicity of a given graph in $O(n^4)$ time via calculating the hyperbolicities of its quadruplets. It is easy to show that the hyperbolicity of a graph is realized on its biconnected component. Thus, for very large graphs, we needed to check hyperbolicities only for quadruplets coming from the same biconnected component. Additionally, we used an algorithm by Cohen et. al. from [27] which has $O(n^4)$ time complexity but performs well in practice as it prunes the search space of quadruplets.

It turns out that most of the quadruplets in our datasets have small δ values (see Table 5). For example, more than 96% of vertex quadruplets in EVA and Erdős datasets have δ values equal to 0. For the remaining graph datasets in Table 5, more than 96% of the quadruplets have $\delta \leq 1$, indicating that all of those graphs are metrically very close to trees.

δ \ Graph	PPI	Yeast	DucthElite	EPA	EVA	California	Erdős
0	0.4831	0.487015	0.54122195	0.5778	0.9973	0.49057007	0.96694
0.5	0.3634	0.450362	0	0.3655	0.0007	0.41052969	0.03278
1	0.1336	0.060844	0.42201697	0.0552	0.0020	0.09527387	0.00028
1.5	0.0179	0.001762	0	0.0015	–	0.00344690	6.80E-08
2	0.0019	0.000017	0.03642388	2.09E-05	–	0.00017945	3.64E-11
2.5	3.55E-05	2.4641E-09	0	1.37E-10	–	0.00000001	–
3	1.65E-06	–	0.00033717	–	–	1.88E-11	–
3.5	3.79E-09	–	0	–	–	–	–
4	–	–	0.00000004	–	–	–	–
% ≤ 1	98.01	99.8221	96.323891	99.84	100	99.637364	99.99999

Table 5: Relative frequency of δ -hyperbolicity of quadruplets in our graph datasets that have less than 10K vertices.

In the remaining part of this section, we discuss the theoretical relations between parameters $\delta(G)$ and $\Delta_s(G)$ of a graph. In [22], the following inequality was proven.

Proposition 2 ([22]). *For every n -vertex graph G and any vertex s of G ,*

$$\Delta_s(G) \leq 4 + 12\delta(G) + 8\delta(G) \log_2 n.$$

Here we complement that inequality by showing that the hyperbolicity of a graph is at most $\Delta_s(G)$.

Proposition 3. *For every n -vertex graph G and any vertex s of G ,*

$$\delta(G) \leq \Delta_s(G).$$

Proof. Let $\mathcal{LP}(G, s)$ be a layering partition of G and $\Gamma(G, s)$ be the corresponding layering tree (consult Fig. 1). From construction of $\mathcal{LP}(G, s)$ and $\Gamma(G, s)$, every cluster C of $\mathcal{LP}(G, s)$ separates in G any two vertices belonging to nodes (clusters) of different subtrees of the forest obtained from $\Gamma(G, s)$ by removing node C . Note that every vertex of G belongs to exactly one node (cluster) of the layering tree $\Gamma(G, s)$.

Consider an arbitrary quadruplet x, y, z, w of vertices of G . Let X, Y, Z, W be the four nodes in $\Gamma(G, s)$ (i.e., four clusters in $\mathcal{LP}(G, s)$) containing vertices x, y, z, w , respectively. In the tree $\Gamma(G, s)$, consider a median node M of nodes X, Y, Z, W , i.e., a node M removing of which from $\Gamma(G, s)$ leaves no connected subtree with more than two nodes from $\{X, Y, Z, W\}$. As a consequence, any connected component of graph $G[V \setminus M]$ (the graph obtained from G by removing vertices of M) cannot have more than 2 vertices out of $\{x, y, z, w\}$. Thus, M separates at least 4 pairs out of the 6 possible pairs formed by vertices x, y, z, w . Assume, without loss of generality, that M separates in G vertices x and y from vertices z and w . See Fig. 2 for an illustration.

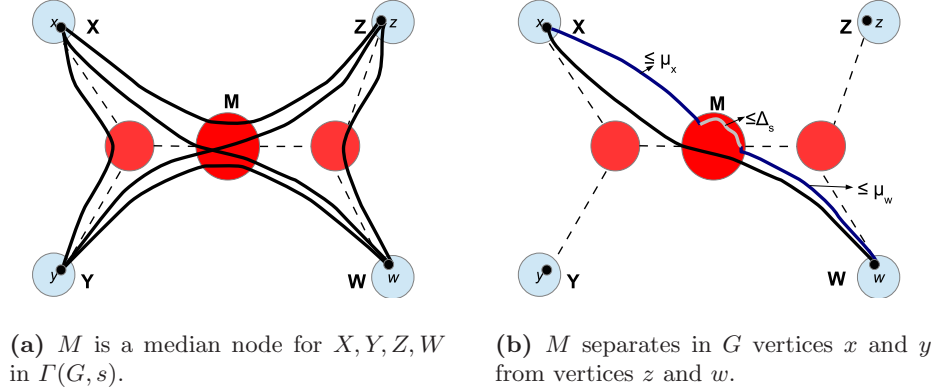


Fig. 2: Illustration to the proof of Proposition 3.

Let μ_a be the distance from $a \in \{x, y, z, w\}$ to its closest vertex in M . Let a, b be a pair of vertices from $\{x, y, z, w\}$. If the vertices a, b belong to different components of $G[V \setminus M]$, then M separates a from b and therefore $\mu_a + \mu_b \leq d_G(a, b)$. Since M separates in G vertices x and y from vertices z and w , we get $d_G(x, z) + d_G(y, w) \geq \mu_x + \mu_y + \mu_z + \mu_w$ and $d_G(x, w) + d_G(y, z) \geq \mu_x + \mu_y + \mu_z + \mu_w$. On the other hand, all three sums $d_G(x, z) + d_G(y, w)$, $d_G(x, w) + d_G(y, z)$ and $d_G(x, y) + d_G(z, w)$ are less than or equal to $\mu_x + \mu_y + \mu_z + \mu_w + 2\Delta_s(G)$, since, by the triangle inequality, $d_G(a, b) \leq \mu_a + \mu_b + \Delta_s(G)$ for every $a, b \in \{x, y, z, w\}$. Now, since the two larger distance sums are between μ and $\mu + 2\Delta_s(G)$, where $\mu := \mu_x + \mu_y + \mu_z + \mu_w$, we conclude that the difference between the two larger distance sums is at most $2\Delta_s(G)$. Thus, necessarily $\delta(G) \leq \Delta_s(G)$. \square

Combining Proposition 2 with Proposition 1, one obtains also the following interesting result relating the hyperbolicity of a graph G with additive distortion of embedding of G to its canonic tree H .

Proposition 4 ([22]). *For any graph $G = (V, E)$ and its canonic tree $H = (V, F)$ the following is true:*

$$\forall u, v \in V, \quad d_H(u, v) - 2 \leq d_G(u, v) \leq d_H(u, v) + O(\delta(G) \log n).$$

Since a canonic tree H is constructible in linear time for a graph G , by Proposition 4, the distances in n -vertex δ -hyperbolic graphs can efficiently be approximated within an additive error of $O(\delta \log n)$ by a tree metric and this approximation is sharp (see [44,43] and [22,41]).

Graphs and general geodesic spaces with small hyperbolicities have many other algorithmic advantages. They allow efficient approximate solutions for a number of optimization problems. For example, Krauthgamer and Lee [53] presented a PTAS for the Traveling Salesman Problem when the set of cities lie in a hyperbolic metric space. Chepoi and Estellon [25] established a relationship between the minimum number of balls of radius $r + 2\delta$ covering a finite subset S of a δ -hyperbolic geodesic space and the size of the maximum r -packing of S and showed how to compute such coverings and packings in polynomial time. Chepoi et al. gave

in [22] efficient algorithms for fast and accurate estimations of diameters and radii of δ -hyperbolic geodesic spaces and graphs. Additionally, Chepoi et al. showed in [23] that every n -vertex δ -hyperbolic graph has an additive $O(\delta \log n)$ -spanner with at most $O(\delta n)$ edges and enjoys an $O(\delta \log n)$ -additive routing labeling scheme with $O(\delta \log^2 n)$ bit labels and $O(\log \delta)$ time routing protocol. We elaborate more on these results in Section 8.

6 Tree-Distortion

The problem of approximating a given graph metric by a “simpler” metric is well motivated from several different perspectives. A particularly simple metric of choice, also favored from the algorithmic point of view, is a tree metric, i.e., a metric arising from shortest path distance on a tree containing the given points. In recent years, a number of authors considered problems of minimum distortion embeddings of graphs into trees (see [5,6,7,24]), most popular among them being a non-contractive embedding with minimum multiplicative distortion.

Let $G = (V, E)$ be a graph. The (multiplicative) *tree-distortion* $td(G)$ of G is the smallest integer α such that G admits a tree (possibly weighted and with Steiner points) with

$$\forall u, v \in V, \quad d_G(u, v) \leq d_T(u, v) \leq \alpha d_G(u, v).$$

The problem of finding, for a given graph G , a tree $T = (V \cup S, F)$ satisfying $d_G(u, v) \leq d_T(u, v) \leq td(G)d_G(u, v)$, for all $u, v \in V$, is known as the *problem of minimum distortion non-contractive embedding of graphs into trees*. In a non-contractive embedding, the distance in the tree must always be larger than or equal to the distance in the graph, i.e., the tree distances “dominate” the graph distances.

It is known that this problem is NP-hard, and even more, the hardness result of [5] implies that it is NP-hard to approximate $td(G)$ better than γ , for some small constant γ . The best known 6-approximation algorithm using layering partition technique was recently given in [24]. It improves the previously known 100-approximation algorithm from [7] and 27-approximation algorithm from [6]. Below we will provide a short description of the method of [24].

The following proposition establishes relationship between the tree-distortion and the cluster-diameter of a graph.

Proposition 5 ([24]). *For every graph G and any its vertex s , $\Delta_s(G)/3 \leq td(G) \leq 2\Delta_s(G) + 2$.*

Proposition 5 shows that the cluster-diameter $\Delta_s(G)$ of a layering partition of a graph G linearly bounds the tree-distortion $td(G)$ of G .

Combining Proposition 5 and Proposition 1, the following result is obtained.

Proposition 6 ([24]). *For any graph $G = (V, E)$ and its canonic tree $H = (V, F)$ the following is true:*

$$\forall u, v \in V, \quad d_H(u, v) - 2 \leq d_G(u, v) \leq d_H(u, v) + 3 td(G).$$

Surprisingly, a multiplicative distortion turned into an additive distortion. Furthermore, while a tree $T = (V \cup S, F)$ satisfying $d_G(u, v) \leq d_T(u, v) \leq td(G)d_G(u, v)$, for all $u, v \in V$, is NP-hard to find, a canonic tree H of G can be constructed in $O(m)$ time (where $m = |E|$).

By assigning proper weights to edges of a canonic tree H or adding at most $n = |V|$ new Steiner points to H , the authors of [24] achieve a good non-contractive embedding of a graph G into a tree. Recall that a canonic tree $H = (V, F)$ of $G = (V, E)$ is constructed in the following way: identify for each cluster $C = L_j^i \in \mathcal{LP}(G, s)$ of a layering partition $\mathcal{LP}(G, s)$ of G an arbitrary vertex $x_C \in L_{i-1}$ which has a neighbor in $C = L_j^i$ and make x_C adjacent in H with all vertices $v \in C$ (see Fig. 3a). Note that H is an unweighted tree, without any Steiner points, and resembles a BFS-tree of G . Two other trees for G are constructed as follows.

Tree H_ℓ : Tree $H_\ell = (V, F, \ell)$ is obtained from H by assigning uniformly the weight $\ell = \max\{d_G(u, v) : uv \text{ is an edge of } H\}$ to all edges of H . So, H_ℓ is a uniformly weighted tree without Steiner points. It turns out that G embeds in tree H_ℓ non-contractively. Note that, although the topology of the tree H_ℓ can be

determined in $O(m)$ time (H_ℓ is isomorphic to H), computation of the weight ℓ requires $O(nm)$ time. Thus, the tree H_ℓ is constructible in $O(nm)$ total time. See Fig. 3a for an illustration.

Tree H'_ℓ : Tree $H'_\ell = (V \cup S, F', \ell)$ is obtained from H by first introducing one Steiner point p_C for each cluster $C := L_j^i$ and adding an edge between each vertex of C and p_C and an edge between p_C and the support vertex x_C for C , and then by assigning uniformly the weight $\ell = \frac{1}{2} \max\{\Delta_s(G), \max\{d_G(u, v) : uv \text{ is an edge of } H\}\}$ to all edges of the obtained tree. So, H'_ℓ is a uniformly weighted tree with at most $O(n)$ Steiner points. Again, G embeds into tree H'_ℓ non-contractively and H'_ℓ can be obtained in $O(nm)$ total time. See Fig. 3b for an illustration.

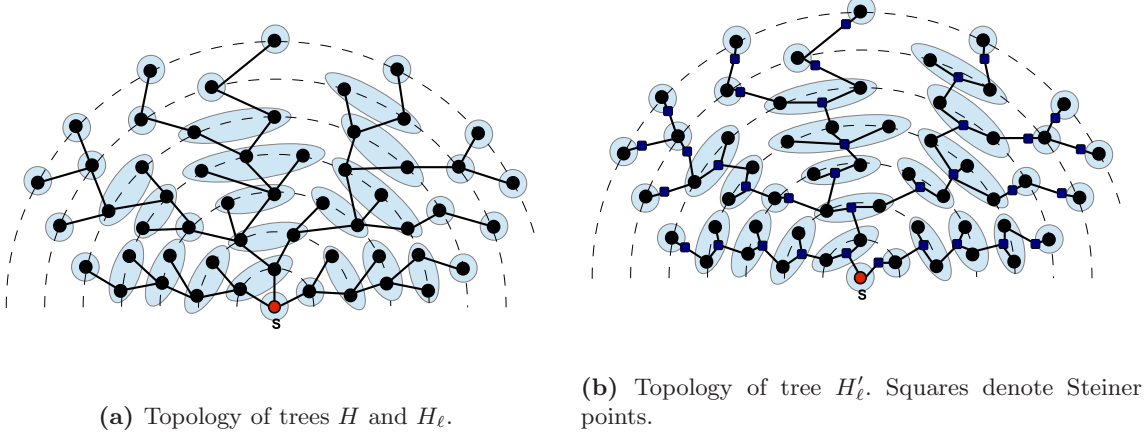


Fig. 3: Embedding into trees H, H_ℓ and H'_ℓ .

Constructed trees have the following distance properties (for comparison reasons, we include also the results for H mentioned earlier).

Proposition 7 ([24]). *Let $G = (V, E)$ be a graph, s be its arbitrary vertex, $\alpha = td(G)$, $\Delta_s = \Delta_s(G)$, and H, H_ℓ, H'_ℓ be trees as described above. Then, for any two vertices x and y of G , the following is true:*

$$\begin{aligned} d_H(x, y) - 2 &\leq d_G(x, y) \leq d_H(x, y) + \Delta_s, \\ d_H(x, y) - 2 &\leq d_G(x, y) \leq d_H(x, y) + 3\alpha, \\ d_G(x, y) &\leq d_{H_\ell}(x, y) \leq (\Delta_s + 1)(d_G(x, y) + 2), \\ d_G(x, y) &\leq d_{H_\ell}(x, y) \leq \max\{3\alpha - 1, 2\alpha + 1\}(d_G(x, y) + 2), \\ d_G(x, y) &\leq d_{H'_\ell}(x, y) \leq (\Delta_s + 1)(d_G(x, y) + 1), \\ d_G(x, y) &\leq d_{H'_\ell}(x, y) \leq 3\alpha(d_G(x, y) + 1). \end{aligned}$$

As pointed out in [24], tree H'_ℓ provides a 6-approximate solution to the problem of minimum distortion non-contractive embedding of graph into tree.

In our empirical study, we analyze embeddings of our graph datasets into each of these three trees and measure how close these graph datasets resemble a tree from this prospective. We compute the following measures:

- *maximum distortion right* $:= \max\{\frac{d_T(u, v)}{d_G(u, v)} : u, v \in V, d_T(u, v) > d_G(u, v) > 0\}$;
- *maximum distortion left* $:= \max\{\frac{d_G(u, v)}{d_T(u, v)} : u, v \in V, d_G(u, v) > d_T(u, v) > 0\}$;
- *average distortion right* $:= \text{avg}\{\frac{d_T(u, v)}{d_G(u, v)} : u, v \in V, d_T(u, v) > d_G(u, v) > 0\}$;
- *average distortion left* $:= \text{avg}\{\frac{d_G(u, v)}{d_T(u, v)} : u, v \in V, d_G(u, v) > d_T(u, v) > 0\}$;

- *average relative distortion* $:= \text{avg}\{\frac{|d_T(u,v)-d_G(u,v)|}{d_G(u,v)} : u, v \in V\}$;
- *distance-weighted average distortion* $:= \frac{1}{\sum_{u,v \in V} d_G(u,v)} \sum_{u,v \in V} (d_G(u,v) \cdot \frac{d_T(u,v)}{d_G(u,v)}) = \frac{\sum_{u,v \in V} d_T(u,v)}{\sum_{u,v \in V} d_G(u,v)}$.

A pair of distinct vertices u, v of $G = (V, E)$ we call a *right pair* with respect to tree $H = (V, F)$ if $d_G(u, v) < d_H(u, v)$. If $d_H(u, v) < d_G(u, v)$ then they are called a *left pair*. Note that G has no left pairs with respect to trees H_ℓ and H'_ℓ , hence, in case of trees H_ℓ and H'_ℓ , we talk only about maximum distortion, average distortion, average relative distortion and distance-weighted average distortion. Distance-weighted average distortion is used in literature when distortion of distant pairs of vertices is more important than that of close pairs, as it gives larger weight values to distortion of distant pairs (see [47]). Clearly, any tree graph would have maximum distortion, average relative distortion and distance-weighted average distortion equal to 1, 0 and 1, respectively.

Tables 6 and 7 show the results of embedding our graph datasets into trees H , H_ℓ and H'_ℓ , respectively. It turns out that most of the datasets embed into tree H with average distortion (right or left, right being usually better) between 1 and 1.5. Also, many pairs of vertices enjoy exact embedding to tree H ; they preserve their original graph distances (for example, around 88% of the pairs in Erdős dataset, 72% of pairs in Homo release 3.2.99, 57% in AS_Caida_20120601 preserve their original graph distances). Comparing the results of non-contractive embeddings to trees H_ℓ and H'_ℓ , we observe that max distortions are slightly improved in H'_ℓ over distortions in H_ℓ , but average distortions are very much comparable. Furthermore, distance-weighted average distortions are better in H_ℓ than in H'_ℓ . This confirms the Gupta's claim in [45] that the Steiner points do not really help.

Graph	average distortion left	max distortion left	% of left pairs (round.)	average distortion right	max distortion right	% of right pairs (round.)	% of pairs $d_T = d_G$ (round.)	average relative distortion	distance-weighted average distortion
PPI	1.50159	7	70.5	1.34140	3	9.1	20.4	0.24669	0.790311
Yeast	1.48714	5	56.3	1.38989	3	12.2	31.5	0.219268	0.850311
DutchElite	1.54045	7	73.0	1.41254	3	3.9	23.1	0.252341	0.760714
EPA	1.50416	5	44.66	1.38107	3	10.47	44.87	0.178557	0.878082
EVA	1.29905	6	32.31	1.27780	3	14.77	52.92	0.110271	0.951626
California	1.52477	5	61.82	1.37071	3	7.92	30.25	0.227176	0.810647
Erdős	1.35242	3	2.75	1.41097	3	8.91	88.34	0.0437277	1.02241
Routeview	1.40636	4	24.39	1.41413	3	33.34	42.28	0.205375	1.03343
Homo release 3.2.99	1.533	4	2.83	1.67827	3	25.16	72.01	0.180092	1.13402
AS_Caida_20071105	1.48085	4	21.43	1.35730	3	35.42	43.15	0.192302	1.02943
Dimes 3/2010	1.53666	3	5.74	1.37247	3	44.42	49.84	0.184767	1.12555
Aqualab 12/2007- 09/2008	1.42269	4	31.71	1.41923	3	35.75	32.54	0.241815	1.03194
AS_Caida_20120601	1.34538	4	22.42	1.40429	3	20.43	57.15	0.138869	1.0068
itdk0304	1.60077	8	94.85	1.26367	3	0.55	4.60	0.331656	0.673012
DBLB-coauth	1.77416	9	95.82	1.24977	3	0.59	3.59	0.383101	0.615328
Amazon	2.48301	19	99.17	1.20027	3	0.20	0.63	0.536656	0.536656

Table 6: Distortion results of embedding datasets into a canonic tree H .

As tree H'_ℓ provides a 6-approximate solution to the problem of minimum distortion non-contractive embedding of graph into tree, dividing by 6 the max distortion values in Table 7 for tree H'_ℓ , we obtain a lower bound on $td(G)$ for each graph dataset G . For example, $td(G)$ is at least $4/3$ for Erdős and Dimes 3/2010, at least $5/3$ for Homo release 3.2.99, at least 2 for Yeast, EPA, Routeview, AS_Caida_20071105, Aqualab 12/2007-09/2008 and AS_Caida_20120601, at least $8/3$ for PPI and California, at least $10/3$ for DutchElite, at least 3 for EVA, at least $11/3$ for itdk0304 and DBLB-coauth, at least 7 for Amazon.

Graph	tree H_ℓ				tree H'_ℓ			
	average distor- tion	max distor- tion	average relative distor- tion	distance- weighted average distor- tion	average distor- tion	max distor- tion	average relative distor- tion	distance- weighted average distor- tion
PPI	5.70566	21	4.70566	5.53218	5.29652	16	4.29652	5.2027
Yeast	4.37781	15	3.37781	4.25155	3.79318	12	2.79318	3.74159
DutchElite	5.45299	21	4.45299	5.325	6.53269	20	5.53269	6.4574
EPA	4.50619	15	3.50619	4.39041	4.06901	12	3.06901	3.99447
EVA	5.83084	18	4.83084	5.70976	7.77752	18	6.77752	7.65544
California	4.15785	15	3.15785	4.05324	4.98668	16	3.98668	4.92935
Erdős	3.08843	9	2.08843	3.06724	3.06705	8	2.06705	3.05622
Routeview	4.28302	12	3.28302	4.13371	4.80363	12	3.80363	4.66503
Homo release 3.2.99	4.64504	12	3.64504	4.53609	3.96703	10	2.96703	3.94713
AS_Caida_20071105	4.24314	12	3.24314	4.11772	4.76795	12	3.76795	4.65617
Dimes 3/2010	3.43833	9	2.43833	3.37664	3.35917	8	2.35917	3.32159
Aqualab 12/2007- 09/2008	4.23183	12	3.23183	4.12775	4.54116	12	3.54116	4.4587
AS_Caida_20120601	4.10547	12	3.10547	4.0272	4.53051	12	3.53051	4.4896
itdk0304	5.370078	24	4.37008	5.3841	5.710122	22	4.71012	5.82908
DBLB-coauth	5.57869	27	4.57869	5.53795	5.12724	22	4.12724	5.14932
Amazon	8.81911	57	7.81911	8.78382	7.87004	42	6.87004	7.95201

Table 7: Distortion results of non-contractive embedding of datasets into trees H_ℓ and H'_ℓ .

7 Tree-Breadth, Tree-Length and Tree-Stretch

There are two other graph parameters measuring metric tree likeness of a graph that are based on the notion of tree-decomposition introduced by Robertson and Seymour in their work on graph minors [60].

A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i | i \in I\}, T = (I, F))$ where $\{X_i | i \in I\}$ is a collection of subsets of V , called *bags*, and T is a tree. The nodes of T are the bags $\{X_i | i \in I\}$ satisfying the following three conditions (see Fig. 4):

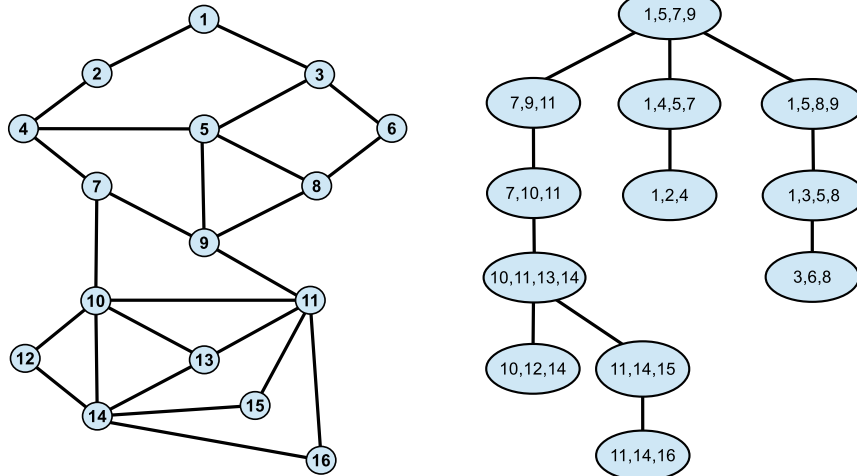
1. $\bigcup_{i \in I} X_i = V$;
2. for each edge $uv \in E$, there is a bag X_i such that $u, v \in X_i$;
3. for all $i, j, k \in I$, if j is on the path from i to k in T , then $X_i \cap X_k \subseteq X_j$. Equivalently, this condition could be stated as follows: for all vertices $v \in V$, the set of bags $\{i \in I | v \in X_i\}$ induces a connected subtree T_v of T .

For simplicity we denote a tree-decomposition $(\{X_i | i \in I\}, T = (I, F))$ of a graph G by $\mathcal{T}(G)$.

The *width* of a tree-decomposition $\mathcal{T}(G) = (\{X_i | i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The *tree-width* of a graph G , denoted by $tw(G)$, is the minimum width over all tree-decompositions $\mathcal{T}(G)$ of G [60]. The trees are exactly the graphs with tree-width 1.

The *length* of a tree-decomposition $\mathcal{T}(G)$ of a graph G is $\lambda := \max_{i \in I} \max_{u, v \in X_i} d_G(u, v)$ (i.e., each bag X_i has diameter at most λ in G). The *tree-length* of G , denoted by $tl(G)$, is the minimum of the length over all tree-decompositions of G [33]. The chordal graphs are exactly the graphs with tree-length 1. Note that these two graph parameters are not related to each other. For instance, a clique on n vertices has tree-length 1 and tree-width $n - 1$, whereas a cycle on $3n$ vertices has tree-width 2 and tree-length n . Analysis of few real-life networks (like Aqualab, AS_Caida, Dimes) performed in [28] shows that although those networks have small hyperbolicities, they all have sufficiently large tree-width due to well connected cores. As we demonstrate below, the tree-length of those graph datasets is relatively small.

The *breadth* of a tree-decomposition $\mathcal{T}(G)$ of a graph G is the minimum integer r such that for every $i \in I$ there is a vertex $v_i \in V$ with $X_i \subseteq B_r(v_i, G)$ (i.e., each bag X_i can be covered by a disk $B_r(v_i, G) := \{u \in V(G) : d_G(u, v_i) \leq r\}$ of radius at most r in G). Note that vertex v_i does not need to belong to X_i . The *tree-breadth* of G , denoted by $tb(G)$, is the minimum of the breadth over all tree-decompositions of G .



(a) A graph G .

(b) A tree-decomposition of G .

Fig. 4: A graph and its tree-decomposition of width 3, of length 3, and of breadth 2.

[36]. Evidently, for any graph G , $1 \leq tb(G) \leq tl(G) \leq 2tb(G)$ holds. Hence, if one parameter is bounded by a constant for a graph G then the other parameter is bounded for G as well.

Clearly, in view of tree-decomposition $\mathcal{T}(G)$ of G , the smaller parameters $tl(G)$ and $tb(G)$ of G are, the closer graph G is to a tree metrically. Unfortunately, while graphs with tree-length 1 (as they are exactly the chordal graphs) can be recognized in linear time, the problem of determining whether a given graph has tree-length at most λ is NP-complete for every fixed $\lambda > 1$ (see [55]). Judging from this result, it is conceivable that the problem of determining whether a given graph has tree-breadth at most ρ is NP-complete, too.

The following proposition from [33] establishes a relationship between the tree-length and the cluster-diameter of a layering partition of a graph.

Proposition 8 ([33]). *For every graph G and any its vertex s , $\Delta_s(G)/3 \leq tl(G) \leq \Delta_s(G) + 1$.*

Thus, the cluster-diameter $\Delta_s(G)$ of a layering partition provides easily computable bounds for the hard to compute parameter $tl(G)$.

One can prove similar inequalities relating the tree-breadth and the cluster-radius of a layering partition of a graph.

Proposition 9. *For every graph G and any its vertex s ,*

$$\Delta_s(G)/6 \leq R_s(G)/3 \leq tb(G) \leq R_s(G) + 1 \leq \Delta_s(G) + 1.$$

Furthermore, a tree-decomposition of G with breadth at most $3tb(G)$ can be constructed in $O(n + m)$ time.

Proof. The proof is similar to the proof from [33] of Proposition 8. First we show $R_s(G)/3 \leq tb(G)$. Let $\mathcal{T}(G)$ be a tree-decomposition of G with minimum breadth $tb(G)$. Let X_1X_2 be an edge of $\mathcal{T}(G)$ and $\mathcal{T}_1, \mathcal{T}_2$ be subtrees of $\mathcal{T}(G)$ after removing the edge X_1X_2 . It is known [30] that set $I = X_1 \cap X_2$ separates in G vertices belonging to bags of \mathcal{T}_1 but not to I from vertices belonging to bags of \mathcal{T}_2 but not to I . Assume that $\mathcal{T}(G)$ is rooted at a bag containing vertex s , the source of layering partition $\mathcal{LP}(G, s)$. Let C be a cluster from layer L_i (i.e., $C = L_i^j$ for some $j = 1, \dots, p_i$). Let Z be the nearest common ancestor of all bags of $\mathcal{T}(G)$ containing vertices of C . Let z be the vertex such that $Z \subseteq B_{tb(G)}(z, G)$.

Consider arbitrary vertex $x \in C$. Necessarily, there is a vertex $y \in C$ and two bags X and Y of $\mathcal{T}(G)$ containing vertices x and y , respectively, such that $Z = NCA_{\mathcal{T}(G)}(X, Y)$ (i.e., Z is the nearest common ancestor of X and Y in $\mathcal{T}(G)$). Let P be a shortest path of G from s to x . By the separator property above, P intersects Z . See Fig. 5 for an illustration. Let a be a vertex of $P \cap Z$ closest to s in G . Since

both x and y belong to C , there exist a path Q from x to y in G using only intermediate vertices w with $d_G(s, w) \geq i$. Let $b \in Q \cap Z$ (i.e. Q intersects Z at vertex b). We have $d_G(s, x) = i = d_G(s, a) + d_G(a, x)$ and $i \leq d_G(s, b) \leq d_G(s, a) + d_G(a, z) + d_G(z, b) \leq d_G(s, a) + 2tb(G)$. Hence, $d_G(a, x) = i - d_G(s, a) \leq 2tb(G)$ and therefore $d_G(x, z) \leq d_G(x, a) + d_G(a, z) \leq 2tb(G) + tb(G) = 3tb(G)$. Thus, any vertex x of C is at distance at most $3tb(G)$ from z in G , implying $R_s(G)/3 \leq tb(G)$.

Note that, for the neighbor x' of x on P , $d(x', z) \leq 3tb(G) - 1$ must hold, i.e., $B_{3tb(G)}(z, G)$ contains not only all vertices of $C = L_i^j$ but also all neighbors of vertices of C laying in layer L_{i-1} . This fact will be useful in the second part of this proof.

Now we show that $tb(G) \leq R_s(G) + 1$. Consider tree $\Gamma(G, s)$ of a layering partition $\mathcal{LP}(G, s)$ and assume $\Gamma(G, s)$ is rooted at node $\{s\}$. Let $p(C)$ be the parent of node C in $\Gamma(G, s)$. Clearly, $\Gamma(G, s)$ satisfies already conditions 1 and 3 of tree-decompositions and only violates condition 2 as the edges joining vertices in different (neighboring) layers are not yet covered by bags (which are the clusters in this case). We can obtain a tree-decomposition Γ' from $\Gamma(G, s)$ as follows. Γ' will have the same structure as $\Gamma(G, s)$, only the nodes of $\Gamma(G, s)$ will slightly expand to cover additional edges of G and form the bags of Γ' . To each node C of $\Gamma(G, s)$ (assume $C \subseteq L_i$) we add all vertices from its parent $p(C)$ ($p(C) \subseteq L_{i-1}$) which are adjacent to vertices of C in G . This expansion of C results in a bag C^+ of Γ' which, by construction, contains now also each edge uv of G with $u \in C \subseteq L_i$ and $v \in p(C) \subseteq L_{i-1}$. Thus, Γ' satisfies conditions 1 and 2 of tree-decompositions. Also, if $C \subseteq B_r(z)$ for

some vertex z and integer r , then $C^+ \subseteq B_{r+1}(z)$ must hold. Furthermore, each vertex v of G that was in a node C now belongs to bag C^+ and to all bags formed from children of C in $\Gamma(G, s)$ (and only to them). Hence, all bags containing v form a star in Γ' . All these indicate that Γ' is a tree-decomposition of G with breadth at most $R_s(G) + 1$, i.e., $tb(G) \leq R_s(G) + 1$.

Furthermore, as we indicated in the first part of this proof, for any cluster C there is a vertex z in G such that $C^+ \subseteq B_{3tb(G)}(z, G)$. The latter implies that the tree Γ' obtained from $\Gamma(G, s)$ has breadth at most $3tb(G)$. Finally, since Γ' is constructible in linear time and $R_s(G) \leq \Delta_s(G) \leq 2R_s(G)$ holds for every graph G , the proposition follows. \square

Hence, the cluster-radius $R_s(G)$ of a layering partition provides easily computable bounds for the tree-breadth $tb(G)$ of a graph. In Table 8, we show the corresponding lower and upper bounds on the tree-breadth for some of our datasets. The lower bound is obtained by dividing $R_s(G)$ by 3, the upper bound is obtained by calculating the breadth of the tree-decomposition Γ' .

Reformulating Proposition 1, we obtain the following result.

Proposition 10. *For any graph $G = (V, E)$ and its canonic tree $H = (V, F)$ the following is true:*

$$\forall u, v \in V, \quad d_H(u, v) - 2 \leq d_G(u, v) \leq d_H(u, v) + 3 \, tl(G) \leq d_H(u, v) + 6 \, tb(G).$$

Graphs with small tree-length or small tree-breadth have many other nice properties. Every n -vertex graph with tree-length $tl(G) = \lambda$ has an additive 2λ -spanner with $O(\lambda n + n \log n)$ edges and an additive 4λ -spanner with $O(\lambda n)$ edges, both constructible in polynomial time [32]. Every n -vertex graph G with $tb(G) = \rho$ has a system of at most $\log_2 n$ collective additive tree $(2\rho \log_2 n)$ -spanners constructible in polynomial time [35]. Those graphs also enjoy a 6λ -additive routing labeling scheme with $O(\lambda \log^2 n)$ bit labels and $O(\log \lambda)$ time routing protocol [31], and a $(2\rho \log_2 n)$ -additive routing labeling scheme with $O(\log^3 n)$ bit labels and $O(1)$ time routing protocol with $O(\log n)$ message initiation time (by combining results of [35] and [37]). See Section 8 for some details.

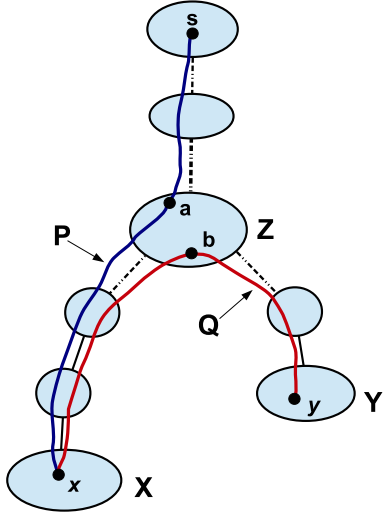


Fig. 5: Illustration to the proof of Proposition 9.

Graph $G = (V, E)$	$R_s(G)$	lower bound on $tb(G)$	upper bound on $tb(G)$
PPI	4	2	5
Yeast	4	2	4
DutchElite	6	2	6
EPA	4	2	4
EVA	5	2	5
California	4	2	4
Erdős	2	1	2
Routeview	3	1	4
Homo release 3.2.99	3	1	3
AS_Caida_20071105	3	1	3
Dimes 3/2010	2	1	2
Aqualab 12/2007- 09/2008	3	1	3
AS_Caida_20120601	3	1	3
itdk0304	6	2	6
DBLB-coauth	7	3	7
Amazon	12	4	12

Table 8: Lower and upper bounds on the tree-breadth of our graph datasets.

Here we elaborate a little bit more on a connection established in [36] between the tree-breadth and the tree-stretch of a graph (and the corresponding tree t -spanner problem).

The *tree-stretch* $ts(G)$ of a graph $G = (V, E)$ is the smallest number t such that G admits a *spanning tree* $T = (V, E')$ with $d_T(u, v) \leq td_G(u, v)$ for every $u, v \in V$. T is called a *tree t -spanner* of G and the problem of finding such tree T for G is known as the *tree t -spanner problem*. Note that as T is a spanning tree of G , necessarily $d_G(u, v) \leq d_T(u, v)$ and $E' \subseteq E$. The latter makes the tree-stretch parameter different from the tree-distortion where new (not from graph) edges can be used to build a tree. It is known that the tree t -spanner problem is NP-hard [15]. The best known approximation algorithms have approximation ratio of $O(\log n)$ [38, 36].

The following two results were obtained in [36].

Proposition 11 ([36]). *For every graph G , $tb(G) \leq \lceil ts(G)/2 \rceil$ and $tl(G) \leq ts(G)$.*

Proposition 12 ([36]). *For every n -vertex graph G , $ts(G) \leq 2tb(G) \log_2 n$. Furthermore, a spanning tree T of G with $d_T(u, v) \leq 2tb(G) \log_2 n d_G(u, v)$, for every $u, v \in V$, can be constructed in polynomial time.*

Proposition 12 is obtained by showing that every n -vertex graph G with $tb(G) = \rho$ admits a tree $(2\rho \log_2 n)$ -spanner constructible in polynomial time. Together with Proposition 11, this provides a $\log_2 n$ -approximate solution for the tree t -spanner problem in general unweighted graphs.

We conclude this section with two other inequalities establishing relations between the tree-stretch and the tree-distortion and hyperbolicity of a graph.

Proposition 13 ([34]). *For every graph G , $tl(G) \leq td(G) \leq ts(G) \leq 2td(G) \log_2 n$.*

Proposition 14 ([34]). *For every δ -hyperbolic graph G , $ts(G) \leq O(\delta \log^2 n)$.*

Proposition 13 says that if a graph G is non-contractively embeddable into a tree with distortion $td(G)$ then it is embeddable into a spanning tree with stretch at most $2td(G) \log_2 n$. Furthermore, a spanning tree with stretch at most $2td(G) \log_2 n$ can be constructed in polynomial time. Proposition 14 says that every δ -hyperbolic graph G admits a tree $O(\delta \log^2 n)$ -spanner. Furthermore, such a spanning tree for a δ -hyperbolic graph can be constructed in polynomial time.

8 Use of Metric Tree-Likeness

As we have mentioned earlier, metric tree-likeness of a graph is useful in a number of ways. Among other advantages, it allows to design compact and efficient approximate distance labeling and routing labeling schemes, fast and accurate estimation of the diameter and the radius of a graph. In this section, we elaborate more on these applications. In general, low distortion embeddability of a graph G into a tree T allows to solve approximately many distance related problems on G by first solving them on the tree T and then interpreting that solution on G .

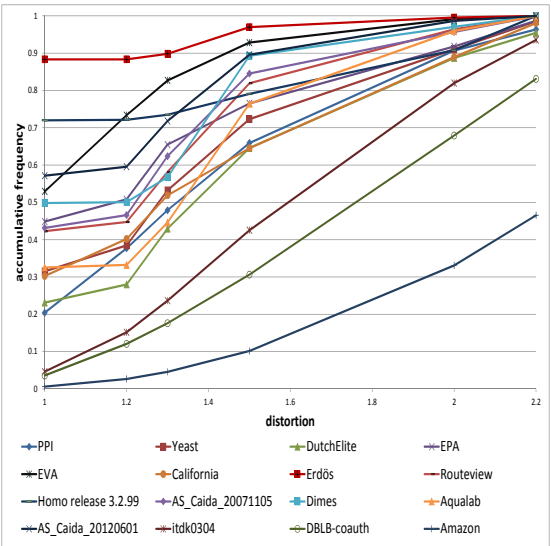
8.1 Approximate distance queries

Commonly, when one makes a query concerning a pair of vertices in a graph (adjacency, distance, shortest route, etc.), one needs to make a global access to the structure storing that information. A compromise to this approach is to store enough information locally in a label associated with a vertex such that the query can be answered using only the information in the labels of two vertices in question and nothing else. Motivation of localized data structure in distributed computing is surveyed and widely discussed in [58,42].

Here, we are mainly interested in the distance and routing labeling schemes, introduced by Peleg (see, e.g., [58]). *Distance labeling schemes* are schemes that label the vertices of a graph with short labels in such a way that the distance between any two vertices u and v can be determined or estimated efficiently by merely inspecting the labels of u and v , without using any other information. *Routing labeling schemes* are schemes that label the vertices of a graph with short labels in such a way that given the label of a source vertex and the label of a destination, it is possible to compute efficiently the port number of the edge from the source that heads in the direction of the destination.

It is known that n -vertex trees enjoy a distance labeling scheme where each vertex is assigned a $O(\log^2 n)$ -bit label such that given labels of two vertices the distance between them can be inferred in constant time [59]. We can use for our datasets their canonic trees to compactly and distributively encode their approximate distance information. Given a graph dataset G , we first compute in linear time its canonic tree H . Then, we preprocess H in $O(n \log n)$ time (see [59]) to assign each vertex $v \in V$ an $O(\log^2 n)$ -bit distance label. Given two vertices $u, v \in V$, we can compute in $O(1)$ time the distance $d_H(u, v)$ from their labels and output this distance as a good estimate for the distance between u and v in G .

Graph $G = (V, E)$	distortion					
	= 1	< 1.2	< 1.3	< 1.5	< 2	< 2.2
PPI	20.41	37.68	47.90	65.93	90.68	96.37
Yeast	31.51	38.45	53.22	72.30	91.03	98.55
DutchElite	23.13	27.99	42.97	64.60	88.71	95.44
EPA	44.87	50.83	65.50	76.52	91.82	98.68
EVA	52.92	73.37	82.68	92.83	99.12	99.88
California	30.25	40.21	51.89	64.53	88.97	98.06
Erdős	88.34	88.34	89.84	96.99	99.55	99.98
Routeview	42.28	44.75	58.17	81.94	96.40	99.85
Homo release 3.2.99	72.01	72.13	73.48	79.08	90.79	99.97
AS_Caida_20071105	43.15	46.60	62.39	84.54	95.68	99.90
Dimes 3/2010	49.84	50.06	56.77	89.30	97.05	99.99
Aqualab 12/2007- 09/2008	32.54	33.23	44.61	76.46	95.93	99.98
AS_Caida_20120601	57.15	59.57	71.82	89.58	98.65	99.98
itdk0304	4.60	15.18	23.67	42.54	81.98	93.55
DBLB-coauth	3.59	12.08	17.60	30.64	67.92	83.10
Amazon	0.63	2.67	4.57	10.16	33.10	46.53



(a) Percentage of vertex pairs whose distance was distorted only up-to a given value.

(b) Accumulative frequency chart.

Fig. 6: Distortion distribution for embedding of a graph dataset into its canonic tree H .

On Fig. 6, we demonstrate how accurate canonic trees represent pairwise distances in our datasets. For a given number $\epsilon \geq 1$, we show how many vertex pairs had a distortion less than ϵ , i.e., pairs $u, v \in V$ with $\max\{\frac{d_H(u,v)}{d_G(u,v)}, \frac{d_G(u,v)}{d_H(u,v)}\} < \epsilon$. We can see that H approximates distances for most vertex pairs with a high level of accuracy. Exact graph distances were preserved in H for at least 40% of pairs in 8 datasets (EPA, EVA, Erdős, Routeview, Homo, AS_Caida_20071105, Dimes 3/2010 and AS_Caida_20120601). At least 50% of pairs of 6 datasets have distance distortion in H less than 1.2. At least 60% of pairs for 6 datasets have distance distortion less than 1.3. At least 70% of pairs of 10 datasets have distance distortion less than 1.5. At least 80% of pairs of 14 datasets have distance distortion less than 2. At least 90% of pairs of 14 datasets have distance distortion less than 2.2. For the DBLB-coauth dataset, 80% (90%) of pairs embed into H with distortion no more than 2.2 (2.4, respectively; not shown on table). For the Amazon dataset, 80% (90%) of pairs embed into H with distortion no more than 3.2 (3.8, respectively; not shown on table).

Hence, using embeddings of our datasets into their canonic trees, we obtain a compact and efficient approximate distance labeling scheme for them. Each vertex of a graph dataset G gets $O(\log^2 n)$ -bit label from the canonic tree and the distance between any two vertices of G can be computed with a good level of accuracy in constant time from their labels only.

8.2 Approximating optimal routes

First we formally define approximate routing labeling schemes. A family \mathfrak{R} of graphs is said to have an $l(n)$ bit (s, r) -approximate routing labeling scheme if there exist a function L , labeling the vertices of each n -vertex graph in \mathfrak{R} with distinct labels of up to $l(n)$ bits, and an efficient algorithm/function f , called the *routing decision* or *routing protocol*, that given the label of a current vertex v and the label of the destination vertex (the header of the packet), decides in time polynomial in the length of the given labels and using only those two labels, whether this packet has already reached its destination, and if not, to which neighbor of v to forward the packet. Furthermore, the routing path from any source s to any destination t produced by this scheme in a graph G from \mathfrak{R} must have the length at most $s \cdot d_G(s, t) + r$. For simplicity, $(1, r)$ -approximate labeling schemes (distance or routing) are called *r -additive labeling schemes*, and $(s, 0)$ -approximate labeling schemes are called *s -multiplicative labeling schemes*.

A very good routing labeling scheme exists for trees [64]. An n -vertex tree can be preprocessed in $O(n \log n)$ time so that each vertex is assigned an $O(\log n)$ -bit routing label. Given the label of a source vertex and the label of a destination, it is possible to compute in constant time the port number of the edge from the source that lays on the (shortest) path to the destination.

Unfortunately, a canonic tree H of a graph G is not suitable for approximately routing in G ; H may have artificial edges (not coming from G) and therefore a path of H from a source to a destination may not be available for routing in G . To reduce the problem of routing in G to routing in a tree T , tree T needs to be a spanning tree of G . Hence, a spanning tree T of G with minimum stretch (i.e., a tree t -spanner of G with $t = ts(G)$) would be a perfect choice. Unfortunately, finding a tree t -spanner of a graph with minimum t is an NP-hard problem.

For our graph datasets, one can exploit the facts that they have small tree-breadth/tree-length and/or small hyperbolicity.

If the tree-breadth of an n -vertex graph G is ρ then, by a result from [36], G admits a tree $(2\rho \log_2 n)$ -spanner constructible in polynomial time. Hence, G enjoys a $(2\rho \log_2 n)$ -multiplicative routing labeling scheme with $O(\log n)$ bit labels and $O(1)$ time routing protocol (routing is essentially done in that tree spanner). Another result for graphs with $tb(G) = \rho$, useful for designing routing labeling schemes, is presented in [35]. It states that every n -vertex graph G with $tb(G) = \rho$ has a system of at most $\log_2 n$ collective additive tree $(2\rho \log_2 n)$ -spanners, i.e., a system \mathcal{T} of at most $\log_2 n$ spanning trees of G such that for any two vertices u, v of G there is a tree T in \mathcal{T} with $d_T(u, v) \leq d_G(u, v) + 2\rho \log_2 n$. Furthermore, such a system \mathcal{T} for G can be constructed in polynomial time [35]. By combining this with a result from [37], we obtain that every n -vertex graph G with $tb(G) = \rho$ enjoys a $(2\rho \log_2 n)$ -additive routing labeling scheme with $O(\log^3 n)$ bit labels and $O(1)$ time routing protocol with $O(\log n)$ message initiation time. The approach of [37] is to assign to each vertex of G a label with $O(\log^3 n)$ bits (distance and routing labels coming from $\log_2 n$ spanning trees) and then, using the label of source vertex v and the label of destination vertex u , identify in $O(\log n)$ time the best spanning tree in \mathcal{T} to route from v to u .

If the tree-length of an n -vertex graph G is λ then, by result from [31], G enjoys a 6λ -additive routing labeling scheme with $O(\lambda \log^2 n)$ bit labels and $O(\log \lambda)$ time routing protocol.

If the hyperbolicity of an n -vertex graph G is δ then, by result from [23], G enjoys an $O(\delta \log n)$ -additive routing labeling scheme with $O(\delta \log^2 n)$ bit labels and $O(\log \delta)$ time routing protocol. Note that for any graph G , the hyperbolicity of G is at most its tree-length [22].

Thus, for our graph datasets, there exists a very compact labeling scheme (at most $O(\log^2 n)$ or $O(\log^3 n)$ bits per vertex) that encodes logarithmic length routes between any pair of vertices, i.e., routes of length at most $d_G(u, v) + \min\{O(\delta \log n), 6\lambda, 2\rho \log_2 n\} \leq \text{diam}(G) + O(\log n) \leq O(\log n)$ for each vertex pair u, v of G . The latter implies very good navigability of our graph datasets. Recall that, for our graph datasets, $\text{diam}(G) \leq O(\log n)$ holds.

Graph $G = (V, E)$	diameter $\text{diam}(G)$	radius $\text{rad}(G)$	# of BFS scans needed to get $\text{diam}(G)$	estimated radius or $\text{ecc}(\cdot)$ of a middle vertex
PPI	19	11	3	12
Yeast	11	6	3	6
DutchElite	22	12	4	13
EPA	10	6	2	7
EVA	18	10	2	10
California	13	7	2	8
Erdős	4	2	2	3
Routeview	10	5	2	5
Homo release 3.2.99	10	5	2	6
AS_Caida_20071105	17	9	2	9
Dimes 3/2010	8	4	2	5
Aqualab 12/2007- 09/2008	9	5	2	5
AS_Caida_20120601	10	5	2	5
itdk0304	26	14	2	15
DBLB-coauth	23	12	2	14
Amazon	47	24	2	26

Table 9: Estimation of diameters and radii.

8.3 Approximating diameter and radius

Recall that the *eccentricity* of a vertex v of a graph G , denoted by $\text{ecc}(v)$, is the maximum distance from v to any other vertex of G , i.e., $\text{ecc}(v) := \max_{u \in V} d_G(v, u)$. The *diameter* $\text{diam}(G)$ of G is the largest eccentricity of a vertex in G , i.e., $\text{diam}(G) := \max_{v \in V} \text{ecc}(v) = \max_{v, u \in V} d_G(u, v)$. The *radius* $\text{rad}(G)$ of G is the smallest eccentricity of a vertex in G , i.e., $\text{rad}(G) := \min_{v \in V} \text{ecc}(v)$. A vertex c of G with $\text{ecc}(c) = \text{rad}(G)$ (i.e., a smallest eccentricity vertex) is called a *central vertex* of G . The *center* $C(G)$ of G is the set of all central vertices of G . Let also $F(v) := \{u \in V : d_G(v, u) = \text{ecc}(v)\}$ be the set of vertices of G furthest from v .

In general (even unweighted) graphs, it is still an open problem whether the diameter and/or the radius of a graph G can be computed faster than the time needed to compute the entire distance matrix of G (which requires $O(nm)$ time for a general unweighted graph). On the other hand, it is known that both, the diameter and the radius, of a tree T can be calculated in linear time. That can be done by using 2 Breadth-First-Search (BFS) scans as follows. Pick an arbitrary vertex u of T . Run a BFS starting from u to find $v \in F(u)$. Run a second BFS starting from v to find $w \in F(v)$. Then $d_T(v, w) = \text{diam}(T)$, i.e., v, w is a *diametral pair* of T , and $\text{rad}(G) = \lfloor (d_T(v, w) + 1)/2 \rfloor$. To find the center of T it suffices to take one or two adjacent middle vertices of the (v, w) -path of T .

Interestingly, in [22], Chepoi et al. established that this approach of 2 BFS-scans can be adapted to provide fast (in linear time) and accurate approximations of the diameter, radius, and center of any finite set S of δ -hyperbolic geodesic spaces and graphs. In particular, for a δ -hyperbolic graph G , it was shown that if $v \in F(u)$ and $w \in F(v)$, then $d_G(v, w) \geq \text{diam}(G) - 2\delta$ and $\text{rad}(G) \leq \lfloor (d_G(v, w) + 1)/2 \rfloor + 3\delta$. Furthermore,

the center $C(G)$ of G is contained in the ball of radius $5\delta + 1$ centered at a middle vertex c of any shortest path connecting v and w in G .

Since our graph datasets have small hyperbolicities, according to [22], few (2, 3, 4, ...) BFS-scans, each next starting at a vertex last visited by the previous scan) should provide a pair of vertices x and y such that $d_G(x, y)$ is close to the diameter $\text{diam}(G)$ of G . Surprisingly (see Table 9), few BFS-scans were sufficient to get exact diameters of all of our datasets: for 13 datasets, 2 BFS-scans (just like for trees) were sufficient to find the exact diameter of a graph. Two datasets needed 3 BFS-scans to find the diameter, and only one dataset required 4 BFS-scans to get the diameter. We also computed the eccentricity of a middle vertex of a longest shortest path produced by these few BFS-scans and reported this eccentricity as an estimation for the graph radius. It turned out that the eccentricity of that middle vertex was equal to the exact radius for 6 datasets, was only one apart from the exact radius for 8 datasets, and only for 2 datasets was two units apart from the exact radius.

9 Conclusion

Based on solid theoretical foundations, we presented strong evidences that a number of real-life networks, taken from different domains like Internet measurements, biological datasets, web graphs, social and collaboration networks, exhibit metric tree-like structures. We investigated a few graph parameters, namely, the tree-distortion and the tree-stretch, the tree-length and the tree-breadth, the Gromov's hyperbolicity, the cluster-diameter and the cluster-radius in a layering partition of a graph, which capture and quantify this phenomenon of being metrically close to a tree. Recent advances in theory allowed us to calculate or accurately estimate these parameters for sufficiently large networks. All these parameters are at most constant or (poly)logarithmic factors apart from each other. Specifically, graph parameters $td(G)$, $tl(G)$, $tb(G)$, $\Delta_s(G)$, $R_s(G)$ are within small constant factors from each other. Parameters $ts(G)$ and $\delta(G)$ are within factor of at most $O(\log n)$ from $td(G)$, $tl(G)$, $tb(G)$, $\Delta_s(G)$, $R_s(G)$. Tree-stretch $ts(G)$ is within factor of at most $O(\log^2 n)$ from hyperbolicity $\delta(G)$. One can summarize those relationships with the following chains of inequalities:

$$\delta(G) \leq \Delta_s(G) \leq O(\delta(G) \log n); \quad R_s(G) \leq \Delta_s(G) \leq 2R_s(G); \quad tb(G) \leq tl(G) \leq 2tb(G);$$

$$\delta(G) \leq tl(G) \leq td(G) \leq ts(G) \leq 2tb(G) \log_2 n \leq O(\delta(G) \log^2 n);$$

$$tl(G) - 1 \leq \Delta_s(G) \leq 3tl(G) \leq 3td(G) \leq 3(2\Delta_s(G) + 2);$$

$$tb(G) - 1 \leq R_s(G) \leq 3tb(G) \leq 3\lceil ts(G)/2 \rceil.$$

If one of these parameters or its average version has small value for a large scale network, we say that that network has a metric tree-like structure. Among these parameters theoretically smallest ones are $\delta(G)$, $R_s(G)$ and $tb(G)$ ($tb(G)$ being at most $R_s(G) + 1$). Our experiments showed that average versions of $\Delta_s(G)$ and of $td(G)$ have also very small values for the investigated graph datasets.

In Table 10, we provide a summary of metric tree-likeness measurements calculated for our datasets. Fig. 7 shows four important metric tree-likeness measurements (scaled) in comparison. Fig. 8 gives pairwise dependencies between those measurements (one as a function of another).

From the experiment results we observe that in almost all cases the measurements seem to be monotonic with respect to each others. The smaller one measurement is for a given dataset, the smaller the other measurements are. There are also a few exceptions. For example, EVA dataset has relatively large cluster-diameter, $\Delta_s(G) = 9$, but small hyperbolicity, $\delta(G) = 1$. On the other hand, Erdős dataset has $\Delta_s(G) = 4$ while its hyperbolicity $\delta(G)$ is equal to 2 (see Figure 8a). Yet Erdős dataset has better embedability (smaller average distortions) to trees H , H_ℓ and H'_ℓ than that of EVA, suggesting that the (average) cluster-diameter may have greater impact on the embedability into trees H , H_ℓ and H'_ℓ . Comparing the measurements of Erdős vs. Homo release 3.2.99, we observe that both have the same hyperbolicity 2, but Erdős has better embedability (average distortion) to trees H , H_ℓ , H'_ℓ . This could be explained by smaller $\Delta_s(G)$ and average diameter of clusters in Erdős dataset. Comparing measurements of PPI vs. California

Graph $G = (V, E)$	diameter $diam(G)$	radius $rad(G)$	cluster- diameter $\Delta_s(G)$	average diameter of clusters in $\mathcal{LP}(G, s)$	$\delta(G)$	Tree H average distortion* (round.)	H_ℓ average distortion	H'_ℓ average distortion	cluster- radius $R_s(G)$
PPI	19	11	8	0.118977384	3.5	1.38471	5.70566	5.29652	4
Yeast	11	6	6	0.119575699	2.5	1.32182	4.37781	3.79318	4
DutchElite	22	12	10	0.070211316	4	1.41056	5.45299	6.53269	6
EPA	10	6	6	0.06698375	2.5	1.26507	4.50619	4.06901	4
EVA	18	10	9	0.031879981	1	1.13766	5.83084	7.77752	5
California	13	7	8	0.092208234	3	1.35380	4.15785	4.98668	4
Erdős	4	2	4	0.001113232	2	1.04630	3.08843	3.06705	2
Routeview	10	5	6	0.063264697	2.5	1.23716	4.28302	4.80363	3
Homo release 3.2.99	10	5	5	0.03432595	2	1.18574	4.64504	3.96703	3
AS_Caida_20071105	17	9	6	0.056424679	2.5	1.22959	4.24314	4.76795	3
Dimes 3/2010	8	4	4	0.056582633	2	1.19626	3.43833	3.35917	2
Aqualab 12/2007- 09/2008	9	5	6	0.05826733	2	1.28390	4.23183	4.54116	3
AS_Caida_20120601	10	5	6	0.055568105	2	1.16005	4.10547	4.53051	3
itdk0304	26	14	11	0.270377048	—	1.57126	5.370078	5.710122	6
DBLB-coauth	23	12	11	0.45350002	—	1.74327	5.57869	5.12724	7
Amazon	47	24	21	0.489056144	—	2.47109	8.81911	7.87004	12

* = $\frac{\text{avg. distortion right} \times \# \text{right pairs} + \text{avg. distortion left} \times \# \text{left pairs} + \# \text{undistorted pairs}}{\binom{n}{2}}$

Table 10: Summary of tree-likeness measurements.

(the same holds for AS_Caida_20071105 vs. AS_Caida_20120601), both have same $\Delta_s(G)$ and $R_s(G)$ values but California (AS_Caida_20120601) has smaller hyperbolicity and average diameter of clusters. We also observe that the datasets Routeview and AS_Caida_20071105 have same values of $\Delta_s(G)$, $R_s(G)$ and $\delta(G)$ but AS_Caida_20071105 has a relatively smaller average diameter of clusters. This could explain why AS_Caida_20071105 has relatively better embedability to H , H_ℓ and H'_ℓ than Routeview. We can see that the difference in average diameters of clusters was relatively small, resulting in small difference in embedability.

From these observations, one can suggest that for classification of our datasets all these tree-likeness measurements are important, they collectively capture and explain metric tree-likeness of them. We suggest that metric tree-likeness measurements in conjunction with other local characteristics of networks, such as the degree distribution and clustering coefficients, provide a more complete unifying picture of networks.

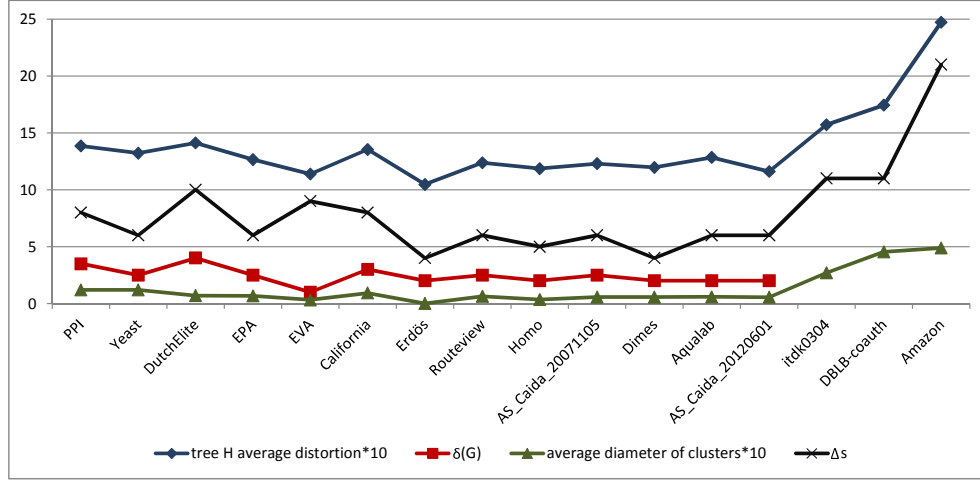


Fig. 7: Four tree-likeness measurements scaled.

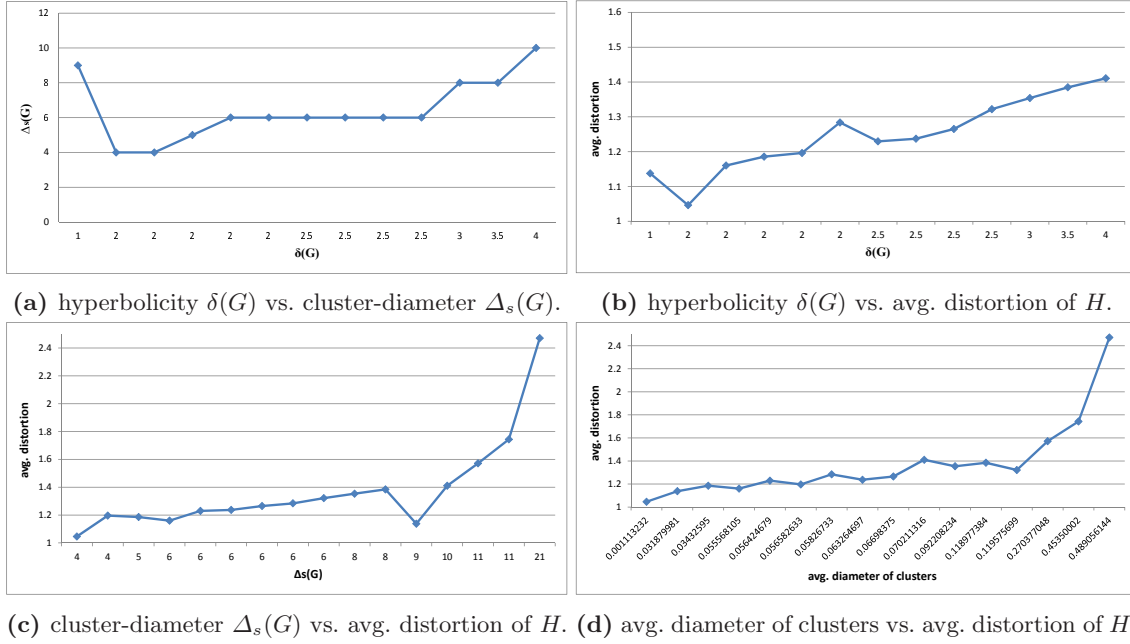


Fig. 8: Tree-likeness measurements: pairwise comparison.

References

1. Pages linking to www.epa.gov. Obtained from Jon Kleinberg’s web page. Available at: <http://www.cs.cornell.edu/courses/cs685/2002fa/>.
2. University of oregon route-views project. <http://www.routeviews.org/>.
3. Ittai Abraham, Mahesh Balakrishnan, Fabian Kuhn, Dahlia Malkhi, Venugopalan Ramasubramanian, and Kunal Talwar. Reconstructing approximate tree metrics. In *PODC*, pages 43–52, 2007.
4. Aaron B. Adcock, Blair D. Sullivan, and Michael W. Mahoney. Tree-like structure in large social and information networks. In *ICDM*, pages 1–10, 2013.
5. Richa Agarwala, Vineet Bafna, Martin Farach, Mike Paterson, and Mikkell Thorup. On the approximability of numerical taxonomy (fitting distances by tree metrics). *SIAM J. Comput.*, 28(3):1073–1085, 1999.
6. Mihai Badoiu, Erik D. Demaine, MohammadTaghi Hajiaghayi, Anastasios Sidiropoulos, and Morteza Zadimoghaddam. Ordinal embedding: Approximation algorithms and dimensionality reduction. In *APPROX-RANDOM*, pages 21–34, 2008.
7. Mihai Badoiu, Piotr Indyk, and Anastasios Sidiropoulos. Approximation algorithms for embedding general metrics into trees. In Nikhil Bansal, Kirk Pruhs, and Clifford Stein, editors, *SODA*, pages 512–521. SIAM, 2007.
8. A. L. Barabasi and R. Albert. Emergence of scaling in random networks. *Science*, 286:509–512, 1999.
9. Albert-László Barabási, Réka Albert, and Hawoong Jeong. Scale-free characteristics of random networks: the topology of the world-wide web. *Physica A: Statistical Mechanics and its Applications*, 281(1-4):69–77, June 2000.
10. Vladimir Batagelj and Andrej Mrvar. Some analyses of Erdos collaboration graph. *Social Networks*, 22(2):173–186, May 2000. <http://vlado.fmf.uni-lj.si/pub/networks/data/Erdos/Erdos02.net>.
11. M. Boguñá, D. Krioukov, and K. C. Claffy. Navigability of complex networks. *Nature Physics*, 5(1):74–80, 2009.
12. Andreas Brandstädt, Victor Chepoi, and Feodor F. Dragan. Distance approximating trees for chordal and dually chordal graphs. *J. Algorithms*, 30(1):166–184, 1999.
13. G. Brinkmann, J. Koolen, and V. Moulton. On the hyperbolicity of chordal graphs. *Annals of Combinatorics*, 5(1):61–69, 2001.
14. Dongbo Bu, Yi Zhao, Lun Cai, Hong Xue, Xiaopeng Zhu, Hongchao Lu, Jingfen Zhang, Shiwei Sun, Lunjiang Ling, Nan Zhang, Guojie Li, and Runsheng Chen. Topological structure analysis of the protein-protein interaction network in budding yeast. *Nucleic Acids Research*, 31(9):2443–2450, May 2003. Available at: <http://vlado.fmf.uni-lj.si/pub/networks/data/bio/Yeast/Yeast.htm>.
15. Leizhen Cai and Derek G. Corneil. Tree spanners. *SIAM J. Discrete Math.*, 8(3):359–387, 1995.
16. CAIDA. The CAIDA AS relationships dataset, 1 June 2012– 5 June 2012. <http://www.caida.org/data/active/as-relationships>.
17. CAIDA. The internet topology data kit #0304, April 2003. <http://www.caida.org/data/active/internet-topology-data-kit>.
18. CAIDA. The CAIDA AS relationships dataset, 5 November 2007. <http://www.caida.org/data/active/as-relationships>.
19. Kai Chen, David R. Choffnes, Rahul Potharaju, Yan Chen, Fabian E. Bustamante, Dan Pei, and Yao Zhao. Where the sidewalk ends: extending the internet as graph using traceroutes from p2p users. In *Proceedings of the 5th international conference on Emerging networking experiments and technologies*, CoNEXT ’09, pages 217–228, New York, NY, USA, 2009. ACM. <http://www.aqualab.cs.northwestern.edu/projects>.
20. Wei Chen, Wenjie Fang, Guangda Hu, and Michael W. Mahoney. On the hyperbolicity of small-world and tree-like random graphs. In Kun-Mao Chao, Tsan sheng Hsu, and Der-Tsai Lee, editors, *ISAAC*, volume 7676 of *Lecture Notes in Computer Science*, pages 278–288. Springer, 2012.
21. Victor Chepoi and Feodor F. Dragan. A note on distance approximating trees in graphs. *Eur. J. Comb.*, 21(6):761–766, 2000.
22. Victor Chepoi, Feodor F. Dragan, Bertrand Estellon, Michel Habib, and Yann Vaxès. Diameters, centers, and approximating trees of delta-hyperbolic geodesic spaces and graphs. In Monique Teillaud, editor, *Symposium on Computational Geometry*, pages 59–68. ACM, 2008.
23. Victor Chepoi, Feodor F. Dragan, Bertrand Estellon, Michel Habib, Yann Vaxès, and Yang Xiang. Additive spanners and distance and routing labeling schemes for hyperbolic graphs. *Algorithmica*, 62(3-4):713–732, 2012.
24. Victor Chepoi, Feodor F. Dragan, Ilan Newman, Yuri Rabinovich, and Yann Vaxès. Constant approximation algorithms for embedding graph metrics into trees and outerplanar graphs. *Discrete & Computational Geometry*, 47(1):187–214, 2012.
25. Victor Chepoi and Bertrand Estellon. Packing and covering δ -hyperbolic spaces by balls. In *APPROX-RANDOM*, pages 59–73, 2007.

26. Fan R. K. Chung and Linyuan Lu. The average distance in a random graph with given expected degrees. *Internet Mathematics*, 1(1):91–113, 2003.
27. Nathann Cohen, David Coudert, and Aurélien Lancin. Exact and approximate algorithms for computing the hyperbolicity of large-scale graphs. Rapport de recherche RR-8074, INRIA, September 2012.
28. Fabien de Montgolfier, Mauricio Soto, and Laurent Viennot. Treewidth and hyperbolicity of the internet. In *NCA*, pages 25–32. IEEE Computer Society, 2011.
29. W. de Nooy. The network data on the administrative elite in the netherlands in April- June 2006. <http://vlado.fmf.uni-lj.si/pub/networks/data/2mode/DutchElite.htm>.
30. Reinhard Diestel. *Graph Theory, 4th Edition*, volume 173 of *Graduate texts in mathematics*. Springer, 2012.
31. Yon Dourisboure. Compact routing schemes for generalised chordal graphs. *J. Graph Algorithms Appl.*, 9(2):277–297, 2005.
32. Yon Dourisboure, Feodor F. Dragan, Cyril Gavoille, and Chenyu Yan. Spanners for bounded tree-length graphs. *Theor. Comput. Sci.*, 383(1):34–44, 2007.
33. Yon Dourisboure and Cyril Gavoille. Tree-decompositions with bags of small diameter. *Discrete Mathematics*, 307(16):2008–2029, 2007.
34. Feodor F. Dragan. Tree-like structures in graphs: a metric point of view. In *WG*, 2013.
35. Feodor F. Dragan and Muad Abu-Ata. Collective additive tree spanners of bounded tree-breadth graphs with generalizations and consequences. In Peter van Emde Boas, Frans C. A. Groen, Giuseppe F. Italiano, Jerzy R. Nawrocki, and Harald Sack, editors, *SOFSEM*, volume 7741 of *Lecture Notes in Computer Science*, pages 194–206. Springer, 2013.
36. Feodor F. Dragan and Ekkehard Köhler. An approximation algorithm for the tree t-spanner problem on unweighted graphs via generalized chordal graphs. In Leslie Ann Goldberg, Klaus Jansen, R. Ravi, and José D. P. Rolim, editors, *APPROX-RANDOM*, volume 6845 of *Lecture Notes in Computer Science*, pages 171–183. Springer, 2011.
37. Feodor F. Dragan, Chenyu Yan, and Derek G. Corneil. Collective tree spanners and routing in at-free related graphs. *J. Graph Algorithms Appl.*, 10(2):97–122, 2006.
38. Yuval Emek and David Peleg. Approximating minimum max-stretch spanning trees on unweighted graphs. *SIAM J. Comput.*, 38(5):1761–1781, 2008.
39. Michalis Faloutsos, Petros Faloutsos, and Christos Faloutsos. On power-law relationships of the internet topology. In *SIGCOMM*, pages 251–262, 1999.
40. Hervé Fournier, Anas Ismail, and Antoine Vigneron. Computing the gromov hyperbolicity of a discrete metric space. *CoRR*, abs/1210.3323, 2012.
41. Cyril Gavoille and Olivier Ly. Distance labeling in hyperbolic graphs. In *ISAAC*, pages 1071–1079, 2005.
42. Cyril Gavoille and David Peleg. Compact and localized distributed data structures. *Distributed Computing*, 16(2-3):111–120, 2003.
43. E. Ghys and P. de la Harpe eds. Les groupes hyperboliques d’après m. gromov. *Progress in Mathematics*, 83, 1990.
44. M Gromov. Hyperbolic groups: Essays in group theory. *MSRI Publ.*, 8:75263, 1987.
45. Anupam Gupta. Steiner points in tree metrics don’t (really) help. In S. Rao Kosaraju, editor, *SODA*, pages 220–227. ACM/SIAM, 2001.
46. H. Jeong, S. P. Mason, A.-L. Barabási, and Z. N. Oltvai. Lethality and centrality in protein networks. *Nature*, 411(6833):41–42, 2001. Available at: <http://www3.nd.edu/~networks/resources.htm>.
47. Mong-Jen Kao, Der-Tsai Lee, and Dorothea Wagner. Approximating metrics by tree metrics of small distance-weighted average stretch. *CoRR*, abs/1301.3252, 2013.
48. W. S. Kennedy, O. Narayan, and I. Saniee. On the Hyperbolicity of Large-Scale Networks. *ArXiv e-prints*, June 2013.
49. Jon M. Kleinberg. Authoritative sources in a hyperlinked environment. *J. ACM*, 46(5):604–632, September 1999. <http://www.cs.cornell.edu/courses/cs685/2002fa/>.
50. Jon M. Kleinberg. The small-world phenomenon: an algorithm perspective. In *STOC*, pages 163–170, 2000.
51. Jon M. Kleinberg. Small-world phenomena and the dynamics of information. In *NIPS*, pages 431–438, 2001.
52. Robert Kleinberg. Geographic routing using hyperbolic space. In *INFOCOM*, pages 1902–1909, 2007.
53. Robert Krauthgamer and James R. Lee. Algorithms on negatively curved spaces. In *FOCS*, pages 119–132, 2006.
54. Jure Leskovec, Kevin J. Lang, Anirban Dasgupta, and Michael W. Mahoney. Community structure in large networks: Natural cluster sizes and the absence of large well-defined clusters. *Internet Mathematics*, 6(1):29–123, 2009.
55. Daniel Lokshtanov. On the complexity of computing treelength. *Discrete Applied Mathematics*, 158(7):820–827, 2010.
56. Onuttom Narayan and Iraj Saniee. Large-scale curvature of networks. *Physical Review E*, 84(6):066108, 2011.

57. K. Norlen, G. Lucas, M. Gebbie, and J. Chuang. EVA: Extraction, Visualization and Analysis of the Telecommunications and Media Ownership Network. *Proceedings of International Telecommunications Society 14th Biennial Conference (ITS2002)*, Seoul Korea, August 2002. Available at: <http://vlado.fmf.uni-lj.si/pub/networks/data/econ/Eva/Eva.htm>.
58. D. Peleg. *Distributed Computing: A Locality-Sensitive Approach*. SIAM Monographs on Discrete Math. Appl. SIAM, Philadelphia, 2000.
59. David Peleg. Proximity-preserving labeling schemes and their applications. In Peter Widmayer, Gabriele Neyer, and Stephan Eidenbenz, editors, *WG*, volume 1665 of *Lecture Notes in Computer Science*, pages 30–41. Springer, 1999.
60. N. Robertson and P. D. Seymour. Graph minors II: algorithmic aspects of tree-width. *Journal Algorithms*, 7:309–322, 1986.
61. Yuval Shavitt and Eran Shir. Dimes: Let the internet measure itself. *CoRR*, abs/cs/0506099, 2005. Available at: <http://www.netdimes.org>.
62. Yuval Shavitt and Tomer Tankel. Hyperbolic embedding of internet graph for distance estimation and overlay construction. *IEEE/ACM Trans. Netw.*, 16(1):25–36, 2008.
63. Chris Stark, Bobby-Joe Breitzkreutz, Teresa Regul, Lorrie Boucher, Ashton Breitzkreutz, and Mike Tyers. Biogrid: a general repository for interaction datasets. *Nucleic Acids Research*, 34(Database-Issue):535–539, 2006. Available at: <http://thebiogrid.org/>, release 3.2.99.
64. Mikkel Thorup and Uri Zwick. Compact routing schemes. In *SPAA*, pages 1–10, 2001.
65. D.J. Watts and S.H. Strogatz. Collective dynamics of 'small-world' networks. *Nature*, (393):440–442, 1998.
66. Yaokun Wu and Chengpeng Zhang. Hyperbolicity and chordality of a graph. *Electr. J. Comb.*, 18(1), 2011.
67. Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In Mohammed Javeed Zaki, Arno Siebes, Jeffrey Xu Yu, Bart Goethals, Geoffrey I. Webb, and Xindong Wu, editors, *ICDM*, pages 745–754. IEEE Computer Society, 2012. Available at: <http://snap.stanford.edu/data/com-Amazon.html>, <http://snap.stanford.edu/data/com-DBLP.html>.